ISS

SECUROS

Версия 11

Руководство программиста

Руководство программиста SecurOS (PG - RU, сборка 97 от 17.08.2024).

Содержание

1 Предисловие

Данный раздел содержит общую информацию о текущем документе, средствах его оформления и использования, а также о порядке получения дополнительной поддержки при эксплуатации продукта.

1.1 Назначение

Данное руководство посвящено программированию системы безопасности SecurOS на языке JavaScript с помощью сценариев, реализуемых в объекте SecurOS *Скрипт Node.js*. В руководстве описываются методы объекта, доступные для использования в сценариях, а также содержатся сведения о событиях и действиях объектов подсистем SecurOS.

Предполагается, что сеть SecurOS и программное обеспечение уже установлены на все компьютеры.

1.2 Целевая аудитория

Руководство предназначено для программистов системы безопасности SecurOS – опытных пользователей компьютера, имеющих навыки по программированию на Node.js и построению сети на основе протокола TCP/IP, знающих основы технологий CCTV.

1.3 Использование руководства

Данный документ можно использовать как в печатном, так и в электронном виде. В последнем случае доступны такие возможности ПО Adobe Reader, как закладки и гипертекстовые ссылки для навигации по документу. Данное руководство ссылается на другие документы по SecurOS, которые можно найти на установочном диске SecurOS или на веб-сайте компании ISS (www.iss.ru).

Данный документ можно вызвать из SecurOS нажатием клавиши **F1**. В режиме администрирования можно открыть статью, посвященную настройкам какого-либо объекта/описанию утилиты, нажав на **F1** при открытом окне настроек данного объекта/окне утилиты. В режиме оператора с помощью клавиши **F1** можно открыть описание активного окна текущего интерфейса оператора или утилиты.

1.4 Обращение за технической поддержкой

При наличии вопросов, ответы на которые отсутствуют в данном руководстве, обратитесь к администратору системы или системному интегратору.

За дальнейшей информацией обращайтесь в Службу технической поддержки компании Intelligent Security Systems:

- По телефону: +7 (495) 645 21 21 (многоканальный, с понедельника по четверг с 9:00 до 18:00, в пятницу с 9:00 до 17:00 по московскому времени);

- Через Портал технической поддержки https://help.iss.ru для оперативной реакции на запрос.

Примечание. Инструкция по работе с Порталом размещена на https://help.iss.ru/ru/site/instruction.

Для более скорого разрешения проблем рекомендуем подготовить служебную информацию, указанную в разделе **Информация для Службы технической поддержки**.

1.5 Соглашение по наименованию редакций SecurOS

Данный документ представляет собой единое руководство для нескольких редакций продукта "Интеграционная платформа видеоменеджмента SecurOS", отличающихся по функциональным возможностям:

- *SecurOS Monitoring & Control Center*;
- *SecurOS Enterprise*;
- *SecurOS Premium*;
- *SecurOS Professional*;
- *SecurOS Xpress*;
- *SecurOS Lite*.

Для обозначения продукта, вне зависимости от его редакции, в рамках данного документа используется единый термин *SecurOS*.

Разделы, в которых описаны функциональности, доступные для некоторых редакций, отмечаются специальной сноской, пример которой представлен ниже:

Функциональность доступна в редакциях *SecurOS Monitoring & Control Center*, *SecurOS Enterprise*, *SecurOS Premium*, *SecurOS Professional*, *SecurOS Xpress*, *SecurOS Lite*.

1.6 Соглашение по оформлению

В данном документе для представления различных терминов и названий используются следующие шрифты и средства форматирования.

Параметр	Описание
жирный	Используется при написании названий рабочих мест, утилит или экранных форм, окон и диалоговых окон, а также названий их элементов.
курсив	Используется для выделения объектов SecurOS.
жирный курсив	Используется для выделения элементов однородных списков.
моноширинный	Используется для выделения текстов макрокоманд и программных кодов, имен файлов и путей к ним. Также используется для указания необходимой опции, выделения значений, задаваемых пользователем с клавиатуры.

Параметр	Описание
зеленый	Используется для выделения перекрестных ссылок внутри документа и ссылок на доступные внешние документы.

1.7 Элементы оформления

Внимание! Служит для привлечения внимания пользователя к информации, которая необходима для корректного восприятия изложенного далее текста. Как правило, данная информация имеет предупреждающий характер.

Примечание. Текст примечания в основном тексте.

Дополнительная информация

Используется для отображения информации дополнительного характера. В элементах такого типа размещается, например, описание вариантов выполнения операции или ссылка на дополнительную литературу.

2 Модуль Node.js

Модуль Node.js (далее Модуль) предназначен для программирования и настройки функционирования сценариев JavaScript в системе SecurOS.

Модуль обеспечивает:

- обработку событий SecurOS;
- управление объектами SecurOS;
- управление конфигурацией объектов SecurOS;
- прочие операции, допустимые при использовании Node.js (операции работы с файловыми системами и сетями).

Внимание! Скрипты Node.js можно использовать как в ОС Windows, так и в ОС Linux.

Перед началом работы начинающим пользователям рекомендуется ознакомится с основами Node.js (см. https://github.com/maxogden/art-of-node#the-art-of-node).

Более детальное изучение Node.js можно начать с API (https://nodejs.org/api/).

2.1 Общие сведения

В разделе рассматриваются следующие вопросы:

- **Краткое введение в Node.js**;
- **Установка ПО Модуля. Особенности реализации**;
- **Настройка и интерфейс Модуля**.

2.1.1 Краткое введение в Node.js

Node.js — это проект с открытым исходным кодом, предназначенный для помощи в написании программ JavaScript, которые взаимодействуют с сетями, файловыми системами или другими источниками ввода/вывода.

В разделе коротко рассматриваются следующие вопросы:

- **Стандарт языка JavaScript (ECMAScript)**;
- **JavaScript в браузерах и Web APIs**;
- **Среда Node.js**;
- **Модули и пакеты**;
- **Синтаксис и библиотека**.

2.1.1.1 Стандарт языка JavaScript (ECMAScript)

Стандартизацией языка JavaScript уже несколько лет занимается организация ECMA, поэтому официально язык называется ECMAScript. Название `JavaScript` продолжает активно использоваться в силу исторических причин. Однако когда речь идёт о версии языка, то тут уже ссылаются именно на спецификацию ECMA: ES5, ES6 и т.д.

Важно сразу определить, что в спецификации языка нет ничего ни про браузер, ни про связь с внешним миром вообще. Язык JavaScript - это лишь синтаксис (типы данных, опеации над ними, операторы и т.д.) и стандартная библиотека, т.е. набор классов/функций для обработки данных и управления логикой программы (работа с числами, строками, датами, форматом JSON, классы коллекций `Array`, `Set` и `Map`, итераторы, буферы памяти и прочее).

2.1.1.2 JavaScript в браузерах и Web APIs

Производители браузеров делают свои реализации JavaScript в соответствии со стандартом ECMA. Вот наиболее известные среди них:

- V8 (Google);
- SpiderMonkey (Mozilla);
- JavaScriptCore (Apple).

Производители браузеров также добавляют интерфейсы, необходимые для взаимодействия с браузером из программы на JavaScript. Эти интерфейсы — просто обычные функции или JavaScript-объекты, которые на момент запуска пользовательской программы уже определены в глобальной области видимости. Например:

- Глобальные функции `setTimeout` и `setInterval` для работы с таймером;
- Объект `document` для чтения/модификации содержимого HTML-страницы;
- Объект `navigator`, служащий для получения свойств браузера, в котором запущена пользовательская программа JavaScript;
- Объект `console` для вывода отладочной информации из программы (в специальное окно);
- и др.

Эти интерфейсы объединяются под общим названием Web API. Они не являются частью языка JavaScript. Их стандартизацией частично занимается организация W3C, частично — сами производители браузеров (если получается достигнуть общего соглашения).

2.1.1.3 Среда Node.js

Node.js представляет собой консольное приложение, в которое встроена одна из реализаций JavaScript (V8). Это приложение позволяет запускать программы, написанные на JavaScript. Обратите внимание, это всё тот же стандартный JavaScript, с обычным синтаксисом и стандартной библиотекой (но без Web API).

Аналогично тому, как это делают производители браузеров, разработчики Node.js также добавили дополнительные библиотеки, позволяющие программе на JavaScript взаимодействовать с внешним миром. Во-первых, они повторили некоторые интерфейсы Web API: глобальные функции `setInterval` и `setTimeout`, глобальный объект `console`, класс `URL` и др. Во-вторых, они добавили свои собственные интерфейсы (модули, пакеты), дающие доступ к файловой системе, сети, оболочке операционной системы и т.п., дополнив всё это некоторыми вспомогательными классами (например, классы `Buffer`, `Stream` и т.п.) и общими библиотеками (криптография, HTTP, работа с архивами, логгирование и проч).

Часть стандартной библиотеки Node.js — это глобальные объекты и функции, которые на момент запуска пользовательской программы уже определены (например, функция `require`, классы `Buffer`, `TextEncoder` и др.). Большая же часть библиотек существует в виде объектов, которые в глобальной области не видны, но на любой их этих объектов можно получить ссылку с помощью глобальной функции `require` (подробнее см. **Модули и пакеты**).

Листинг 1. Типичные переменные

```
// Типичный подход - когда все необходимые библиотеки мы получаем заранее
// и сохраняем в переменных:
const fs = require('fs');
const net = require('net');
fs.readFile(...);
net.connect(...)
```

2.1.1.4 Модули и пакеты

Модуль представляет собой отдельный JavaScript-файл. Чаще подразумевается JavaScript-файл, содержащий собой набор каких-то функций, классов и объектов (переменных, констант), предназначенных для решения отдельной небольшой задачи. Часть этих функций, классов экспортируются. Для этого в коде модуля делается сохранение ссылок в специальном объекте `module.exports`. Оставшаяся часть функций и объектов, связанных с деталями реализации модуля, остаётся недоступной (скрытой внутри модуля). В любом другом месте кода (в коде скрипта, в любом другом модуле) можно получить доступ к экспорту с помощью функции `require`. Для каждого модуля на этапе его загрузки создаётся свой объект `module.exports`. Загрузка любого модуля выполняется лишь один раз, при первом вызове `require` для этого модуля. Все последующие вызовы require будут возвращать копию уже созданного объекта `module.exports`.

Пакет представляет собой набор из нескольких модулей и специального файла `package.json`, которые располагаются в отдельном каталоге (внутри каталога `node_modules`). У любого пакета всегда есть главный модуль (какой именно — описывается в файле `package.json`). Импорт пакета, по сути, представляет собой импорт именно главного модуля пакета.

Эко-система Node.js выгодно отличается наличием централизованного публичного репозитория `registry.npmjs.org`, в котором есть огромное количество пакетов под разные задачи (см. **Установка и использование дополнительных пакетов**). Для поиска подходящих пакетов существует специальный сервис https://www.npmjs.com/. Установка пакетов осуществляется с помощью консольной утилиты `npm` (Node Package Manager).

2.1.1.5 Синтаксис и библиотека

В разделе перечислены некоторые новые возможности языка, появившиеся в стандартах ES6 и ES7:

- **Объявление переменных**;
- **Запись циклов с помощью итерируемого объекта**;
- **Шаблонные строки**;
- **Promise API**;
- **Асинхронные функции**.

2.1.1.5.1 Объявление переменных

Кроме ключевого слова `var`, переменные можно объявить с помощью ключевых слов `let` и `const`.

У переменных, объявленных с помощью ключевого слова `var`, область видимости распространяется на всю функцию. Взгляните на следующий пример:

```
function test() {
    ...

    if (...) {
        var x = 12;
        ...
    }
}
```

Кажется, что переменная `x` существует только внутри блока `if`. Однако это не так! Область видимости переменной — вся функция `test`. Поэтому в данном примере переменная `x` на самом деле ведёт себя так:

```
function test() {
    var x;
    ...
    // к 'x' можно обращаться здесь
    if (...) {
        x = 12;
        ...
        // и здесь
    }
    // и здесь тоже
}
```

Такое поведение служит частым источником ошибок. Это проблема в дизайне языка, которую, к сожалению, исправить уже нельзя, т.к. это вызовет нарушение совместимости со старым кодом. Поэтому в стандарте ES6 появилось новое ключевое слово `let`. Оно имеет тот же смысл, что и `var`, однако область видимости таких переменных теперь уже действительно ограничивается только блоком:

```
function test() {
    ...

    if (...) {
        let x = 12; // теперь 'x' видна только в этом блоке,
                    // как и ожидается

        ...
    }
}
```

Ключевое слово `const` — еще один способ объявления переменных. Область видимости такой переменной так же ограничивается блоком. Но, в отличие от `let`, переменная `const` инициализируется только один раз. При попытке "перепривязать" её к какому-то другому объекту (числу, строке, массиву `Array`, функция и проч.) возникнет ошибка на этапе исполнения:

```
const x = 12;       // переменная 'x' держит ссылку на объект типа
                    // 'number' со значением 12
const s = "12345"; // переменная 's' держит ссылку на объект типа
                    // 'string' со значением "12345"
const obj = {       // переменная 'obj' держит ссылку на объект типа
                    // 'object' со свойствами 'a' и 'b'
    a: 12,
    b: true
};
...

x = 11;       // ОШИБКА! пытаемся привязать переменную 'x' к другому
              // объекту типа 'number' со значением 11
s = "54321"; // ОШИБКА! пытаемся привязать переменную 's' к другому
              // объекту типа 'string' со значением "54321"

obj.a = 100; // разрешается, переменная 'obj' всё ещё ссылается на тот же
              // объект, но мы можем менять свойства объекта,
              // вызывать его методы и т.д.
obj = {       // ОШИБКА! мы пытаемся привязать переменную 'obj' к другому
              // объекту
    a: 12,
    b: true
};
```

Объявление переменных с помощью `const` помогает защищаться от логических ошибок, а именно: помогает обнаруживать ситуации, когда мы случайно поменяли что-то, что меняться не должно. Ошибка, выданная интерпретатором на этапе исполнения, с номером строки и подробным описанием — это гораздо лучше, чем "странное непонятное" поведение программы.

2.1.1.5.2 Запись циклов с помощью итерируемого объекта

В JavaScript появилось понятие **итерируемого объекта** — обобщённый интерфейс для перебора всех элементов какого либо контейнера:

```
    for (elem of iterable) {
        ...
    }
```

Здесь `iterable` обозначает любой итерируемый объект. Примерами таких объектов являются массивы (`Array`), новые типы контейнеров `Map` и `Set`, а также специальные функции-генераторы (специальный тип функции, которые на каждом вызове возвращает какое-то новое значение). Поэтому, например, перебор всех элементов массива теперь можно записать двумя способами:

```
let arr = [1, 2, 'a', 'b', 'c'];
```

```
// способ 1 - через индексы
for (let i = 0; i < arr.length; ++i) {
    let elem = arr[i];
    ...
}

// способ 2 - через интерфейс итераторов
for (let elem of arr) {
    ...
}
```

При этом в большинстве случаев цикл `for .. of` будет короче при записи и удобнее для использования.

2.1.1.5.3 Шаблонные строки

Шаблонные строки (литералы) представляют собой строки, внутри которых можно использовать выражения:

```
const name = "Олег";
const height = 200;
function getStatus() { return "весёлый"; }

const message = `Меня зовут ${name}, мой рост ${height} см,
сегодня я ${getStatus()}`;
// в переменной 'message' будет строка "Меня зовут Олег,
// мой рост 200 см, сегодня я весёлый"
```

Строка ограничивается символами ` (обратный апостроф), а внутрь можно вставлять выражения вида `${выражение}`. Выражением может любое корректное выражение, значение которого может быть преобразовано в строку: имя переменной, вызов функции, число, строка и т.д. Такой способ удобнее, чем традиционная конкатенация строки из нескольких частей (`"Меня зовут " + name + ", мой рост " + height + " см"`).

2.1.1.5.4 Promise API

В стандартной библиотеке ES6 появился новый класс `Promise`, который является оберткой над асинхронной операцией. Под асинхронной понимается некая операция, которая выполняется "параллельно" и завершается, как правило, вызовом callback-функции, в которую передаётся результат операции (либо вызов callback-функции сам по себе служит сигналом окончания операции). И, хотя в стандартной библиотеке JavaScript асинхронных операций нет, однако они есть в браузерах или в библиотеке Node.js. Например:

- Посылка HTTP-запроса (типичная операция в браузерах) - через callback Вы, например, получаете ответ HTTP-сервера;
- Чтение/запись файла (через функции пакета `fs` в Node.js) - через callback Вы, например, получаете блок считанных данных;
- и т.д.

Чаще всего при запуске асинхронной операции ей передаётся либо единственная callback-функция для получения результата, либо две callback-функции - для получения результата и для обработки ошибки.

Обратите внимание, что не всё подпадает под определение асинхронных операций. Например, при регистрации обработчика какого-либо события (щелчок мышью в браузере, приём очередного блока данных из TCP-соединения и т.п.), то здесь применение интерфейса `Promise` не подходит вследствие того, что:

- Никакая операция не запускается. Просто регистрируется обработчик какого-либо события;
- Событие, возможно, когда-то возникнет, может быть даже несколько раз, а может не возникнуть никогда.

Таким образом, интерфейс `Promise` используется для асинхронных операций, которые:

- Запускаются сразу же;
- Выполняются только один раз;
- Завершаются (возможно, с каким-то результатом).

Теперь, чтобы понять какие именно проблемы решает новый класс `Promise`, рассмотрим следующий искусственный пример. Допустим, у нас есть три асинхронные операции:

1. Чтение из файла;
2. Обработка данных;
3. Отправка данных на сервер.

```
// асинхронное чтение из файла, содержимое возвращается через функцию
// обратного вызова 'callback'
function readFile(path, callback) { ... }

// асинхронная обработка данных, результат возвращает через функцию
// обратного вызова 'callback'
function process(data, callback) { ... }

// асинхронная отправка данных на сервер, результат (успех/не успех)
// возвращается через функцию обратного вызова 'callback'
function upload(url, data, callback) { ... }
```

Основная проблема возникает в том случае, когда эти операции необходимо выполнить последовательно, друг за другом:

```
// запускаем цепочку операций
readFile('my-file.txt', function (content) {
    // ... что-то делаем
    process(content, function(result) {
        // .. что-то делаем
        upload('my-server.com', result, function(success) {
            // проверяем success, возможно, что-то делаем
        });
    });
});
```

Как видите, с увеличением шагов в цепочке операций код постепенно смещается вправо, становится менее читаемым и менее понятным. В попытке оформить все callback-функции отдельно результат будет только хуже. Код теперь не смещается вправо, но понять его логику становится явно сложнее:

```
function onReadFile(content) {
    // ... что-то делаем
    process(content, onProcessComplete);
}

function onProcessComplete(result) {
    // .. что-то делаем
    upload('my-server.com', result, onUploadComplete);
}

function onUploadComplete(success) {
    // что-то делаем
}

// запускаем цепочку операций
readFile('my-file.txt', onReadFile);
```

Теперь посмотрим, что предлагает нам класс `Promise`. При создании объекта `Promise` ему передается функция (executor), которая принимает два callback-обработчика: обработчик результата и обработчик ошибки:

```
let promise = new Promise(function(resolve, error) {
    // здесь запускается какая-то асинхронная операция
    // при успешном завершении будет вызвана функция resolve
    // в случае ошибки можно вызвать функцию error, но это опционально
});
```

Теперь, чтобы получить результат асинхронной операции, нужно вызвать метод `then` объекта `Promise`, передав ему callback-функцию, принимающую результат:

```
// асинхронная операция '1'
let promise = new Promise(function(resolve, error) { ...});

promise.then(function(result) {
    // наш обработчик, будет вызван, когда асинхронная операция '1'
    // выполнится успешно
});
```

Метод `then` возвращает новый объект `Promise`, используя который мы можем получить результат вызова обработчика. Обработчик может вернуть какое-то значение (число, строку, объект, массив и т.д.):

```
// асинхронная операция '1'
let promise = new Promise(function(resolve, error) { ...});

let promise_2 = promise.then(function(result) {
    // наш обработчик, будет вызван, когда асинхронная операция '1'
    // выполнится успешно
    // что-то делаем и что-то возвращаем
    return [1, 2, 'ok'];
});
```

```
promise_2.then(function(result) {
    // обработчик будет вызван, после того, как выполнится асинхронная
    // операция '1' и отработает предыдущий обработчик
    // result будет содержать [1, 2, 'ok']
});
```

Можно добавить в метод `then` обработчик, который при вызове запустит новую асинхронную операцию и вернёт ее в виде объекта `Promise`. Тогда у нас получится цепочка асинхронных операций:

```
// асинхронная операция '1'
let promise = new Promise(function(resolve, error) { ...});

let promise_2 = promise.then(function(result) {
    // наш обработчик, будет вызван, когда асинхронная операция '1'
    // выполнится успешно
    // мы запускаем новую асинхронную операцию '2'
    return new Promise(...)
});

let promise_3 = promise_2.then(function(result) {
    // обработчик будет вызван, когда асинхронные операции '1' и '2'
    // будут выполнены успешно, по цепочке
    // мы запускаем новую асинхронную операцию '3'
    return new Promise(...)
});

let promise_4 = promise_3.then(function(result) {
    // обработчик будет вызван, когда асинхронные операции '1', '2' и '3'
    // будут выполнены успешно, по цепочке
    // мы запускаем новую асинхронную операцию '4'
    return new Promise(...)
});

// ... и т.д.
```

Как правило, нет необходимости сохранять промежуточные объекты `Promise`, поэтому то же самое можно записать в более короткой форме:

```
// Каждый следующий вызов then - это вызов метода нового объекта Promise,
// созданного на предыдущем шаге then(...)
new Promise(function(resolve, error) { ...})
    .then(function(result) {
        ...
        return new Promise(...)
    })
    .then(function(result) {
        ...
        return new Promise(...)
    })
    .then(function(result) {
        ...
        return new Promise(...)
    });
```

```
    .then(function(result) {
        // ... и т.д.
    });
```

Давайте перепишем наш исходный пример с использованием `Promise`. Сначала напишем обёртки над нашими асинхронными операциями, затем вызовем их по цепочке:

```
// асинхронное чтение из файла, содержимое возвращается через Promise
function readFilePromise(path) {
    return new Promise(function(resolve, error) {
        readFile(path, resolve);
    });
}

// асинхронная обработка данных, результат возвращает через Promise
function processPromise(data) {
    return new Promise(function(resolve, error) {
        process(data, resolve);
    })
}

// асинхронная отправка данных на сервер, результат (успех/не успех)
// возвращается через Promise
function uploadPromise(url, data) {
    return new Promise(function(resolve, error) {
        upload(url, data, resolve);
    })
}

// запускаем цепочку операций
readFilePromise('my-file.txt')
    .then(function(content) {
        // ... что-то делаем
        return processPromise(content);
    })
    .then(function(result) {
        // .. что-то делаем
        return uploadPromise('my-server.com', result);
    })
    .then(function(success) {
        // проверяем success, возможно что-то делаем
    });
```

Как видно, теперь код не смещается вправо, при этом логика воспринимается очень легко, а значит такой код меньше подвержен ошибкам и его проще модифицировать. В браузерах все новые интерфейсы теперь делаются на основе `Promise`. В библиотеках Node.js часть интерфейсов уже сделана на объектах `Promise` (например, `fs`). Для всех остальных случаев в пакете `util` есть функция `promisify`. С помощью этой функции можно обернуть в объект `Promise` практически любой библиотечный асинхронный вызов.

2.1.1.5.5 Асинхронные функции

В стандарте ES7 появилась поддержка объектов `Promise` (асинхронных операций) на уровне синтаксиса:

- асинхронные функции — специальный тип функций, объявляемых с ключевым словом `async`;
- оператор `await`, позволяющий получить результат асинхронной функции. Асинхронные функции — специальный тип функций, которые всегда возвращают объект `Promise`. Объект `Promise` можно вернуть из такой функции в явном виде. Любые другие явно возвращаемые будут обёрнуты в объект `Promise`. Если же в коде функции нет возвращаемого значения в явном виде (`return`), всё, что выполняется внутри такой функции, оборачивается в `Promise`, который и возвращается из неё неявно.

```
// обычная функция
function func() {
    return "12345";
}

let a = func(); // в переменной 'a' будет строка

// асинхронная функция
async function asyncFunc() {
    return "12345";
}

// в переменной 'b' будет объект Promise
let b = asyncFunc();
b.then(function(result) {
    // в 'result' будет строка "12345"
});
```

Внутри асинхронной функции также можно выполнять какие-то асинхронные операции. При этом результат асинхронной операции можно получить с помощью оператора `await`:

```
// функция запускает некую асинхронную операцию и возвращает Promise
function run() { ... }


// способ первый
function doRun() {
    return run().then(function(result) {
        // обрабатываем 'result' и возвращаем некий конечный результат 'x'
        ...
        return x;
    });
}

doRun().then(function(x) {
    // будет вызвана по окончании асинхронной операции
});


// способ второй - с использованием async/await (по сути то же самое)
async function doRunAsync() {
    let result = await run();
```

```
    // обрабатываем 'result' и возвращаем некий конечный результат 'x'
    ...
    return x;
}

doRunAsync().then(function(x) {
    // будет вызвана по окончании асинхронной операции
});
```

Давайте перепишем пример из раздела **Promise API**. Сначала оформим всю цепочку в виде обычной функции, возвращающей `Promise`:

```
function doOperation() {
    return readFilePromise('my-file.txt')
        .then(function(content) {
            // ... что-то делаем
            return processPromise(content);
        })
        .then(function(result) {
            // .. что-то делаем
            return uploadPromise('my-server.com', result);
        })
        .then(function(success) {
            // проверяем success, возможно что-то делаем
        });
}

// можно просто запустить цепочку операции
doOperation();

// или запустить и добавить callback-функцию, которая будет вызвана
// после завершения всей цепочки
doOperation()
    .then(function() {
        // вся цепочка выполнена
    });
```

Теперь сделаем то же самое с помощью асинхронной функции и оператора `await`:

```
async function doOperationAsync() {
    const content = await readFilePromise('my-file.txt');
    // ... что-то делаем
    const result = await processPromise(content);
    // .. что-то делаем
    const success = await uploadPromise('my-server.com', result);
    // проверяем success, возможно, что-то делаем
}

// аналогично можно просто запустить цепочку операции
doOperationAsync();

// или запустить и добавить callback-функцию, которая будет вызвана
// после завершения всей цепочки
doOperationAsync()
    .then(function() {
```

```
        // вся цепочка выполнена
    });
```

Эти два примера эквиваленты. Однако последний вариант с асинхронной функцией и оператором `await` выглядит компактнее и проще.

Обратите внимание на следующее:

1. вызов асинхронной функции не обязательно означает, что все инструкции в её телё уже выполнены (возможно, они пока лишь запланированы на исполнение в будущем);
2. оператор `await` можно использовать только внутри асинхронных функций;
3. `await` по-сути блокирует только исполнение инструкций в теле асинхронной функции;
4. при этом сам код, из которого была вызвана асинхронная функция, никогда не блокируется.

2.1.2 Установка ПО Модуля. Особенности реализации

Модуль может быть установлен как в ОС Windows, так и в ОС Linux. Подробности установки и структуры каталогов файловой системы для каждой ОС описаны в следующих разделах:

- **Особенности реализации на Windows**;
- **Особенности реализации на Linux**.

2.1.2.1 Особенности реализации на Windows

Бинарный файл *Node.js* (`node.exe`) поставляется вместе с инсталлятором. Структура каталогов выглядит следующим образом:

- `C:/Program Files (x86)/ISS/SecurOS/bin64`
  - ... модули SecurOS
  - `nodejs.exe` — исполнитель
  - `[node.js]` — рабочий каталог при запуске скрипта
    - `[node_modules]` — пакеты Node.js (размещайте свои пакеты здесь)
      - `[securos]` — пакет для взаимодействия с ядром
      - `[npm]` — каталог модулей менеджера пакетов `npm`
      - ... другие пакеты (устанавливаются пользователем)
    - `node.exe` — исполняемый файл внешнего приложения node, версия 12.18.3

2.1.2.2 Особенности реализации на Linux

На ОС Linux среда node устанавливается глобально системным менеджером пакетов из публичного репозитория. Обратите внимание, что версия Node.js зависит от используемого дистрибутива ОС:

- Astra Linux — `node v.8`;
- Debian — `node v.10`.

Структура каталогов выглядит следующим образом:

- `/opt/iss/SecurOS/bin64`
  - ... модули SecurOS
  - `nodejs` — исполнитель
  - `[node.js]` — рабочий каталог при запуске скрипта
    - `[node_modules]` — пакеты Node.js (размещайте свои пакеты здесь)
      - `[securos]` — пакет для взаимодействия с ядром
      - ... другие пакеты (устанавливаются пользователем)

2.1.3 Настройка и интерфейс Модуля

В данном разделе приводится описание интерфейса Модуля и параметров его настройки:

- **Создание и настройка объекта Скрипты Node.js**;
- **Создание и настройка объекта Скрипт Node.js**.

2.1.3.1 Создание и настройка объекта Скрипты Node.js

Данный объект не имеет настраиваемых параметров и служит для группировки рабочих объектов Модуля.

Родительский объект — *Компьютер*/группа *Интеграция и Автоматизация*.

Для создания и настройки объекта *Скрипты Node.js* выполните следующие действия:

1. Войдите в режим администрирования.
2. В дереве объектов SecurOS выберите объект *Компьютер*, для которого в группе *Интеграция и Автоматизация* создайте дочерний объект *Скрипты Node.js*. В окне **Параметры создаваемого объекта** (здесь и далее не приводится) задайте требуемые значения.
3. Примените новые настройки.

2.1.3.2 Создание и настройка объекта Скрипт Node.js

Данный объект реализует интерфейс настройки сценария и одновременно является средой разработки сценария.

Родительский объект — *Компьютер*/группа *Интеграция и Автоматизация*/*Скрипты Node.js*.

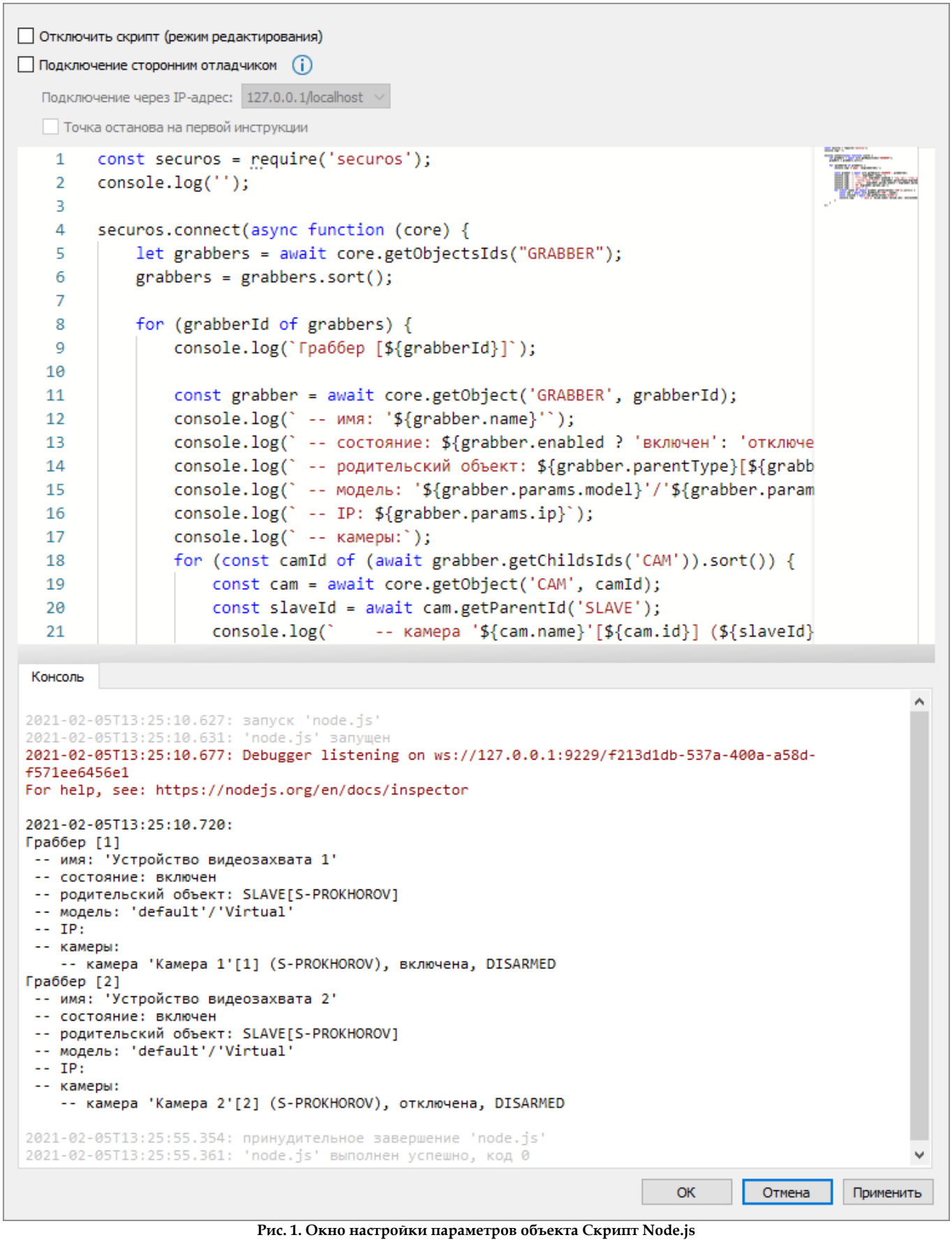

Рис. 1. Окно настройки параметров объекта Скрипт Node.js

Для создания и настройки объекта *Скрипт Node.js* выполните следующие действия:

1. Войдите в режим администрирования.

2. В дереве объектов SecurOS выберите объект *Скрипты Node.js*, для которого создайте дочерний объект *Скрипт Node.js*. В окне **Параметры создаваемого объекта** (здесь и далее не приводится) задайте требуемые значения.
3. В окне настройки объекта задайте значения параметров (см. ниже).

Таблица 1. Параметры объекта Скрипт Node.js

Параметр	Описание
Отключить скрипт (режим редактирования)	Флаг включения режима редактирования скрипта. В данном режиме скрипт выполняться не будет. По умолчанию не отмечен.
Подключение сторонним отладчиком	Флаг возможности подключения произвольного внешнего отладчика. По умолчанию не отмечен. Параметр недоступен в следующих случаях: • если отмечен флаг **Отключить скрипт (режим редактирования)**; • не удалось получить список IP-адресов для подключения (см. параметр **Подключение через IP-адрес**).
Подключение через IP-адрес	Содержит список внешних IP-адресов того сервера, на котором расположен тестируемый скрипт (объект *Скрипт Node.js*). Для подключения внешним отладчиком возможно использовать любой из IP-адресов списка. Если в списке выбрано какое-либо значение, это же значение необходимо указать в настройках отладчика при его запуске. **Примечание.** Если выбрано значение `localhost`, подключение внешним отладчиком возможно только с того *Компьютера*, где создан скрипт. Параметр недоступен, если выполняется хотя бы одно из следующих условий: • отмечен флаг **Отключить скрипт (режим редактирования)**; • не отмечен флаг **Подключение сторонним отладчиком**; • не удалось получить список IP-адресов для подключения (например, при отсутствии соединения).
Точка останова на первой инструкции	Флаг ожидания подключения отладчиком. Доступен в случае, если отмечен флаг **Подключение сторонним отладчиком**. Если не отмечен, скрипт будет выполняться непрерывно.
Окно сценария	Текстовое поле, содержащее код скрипта. **Примечание.** Для написания сценария используется встроенный редактор **Monaco editor**. Подробное описание редактора можно найти на официальном сайте проекта https://microsoft.github.io/monaco-editor/.

Параметр	Описание
Окно результата выполнения сценария (Консоль)	Содержит данные стандартных потоков `stdout` и `stderr`, соответствующие примененному, а не отредактированному скрипту. Результат работы скрипта или детальное сообщение об ошибке будут выводиться в окне при работе скрипта.

4. Создайте сценарий (см. **Создание и выполнение сценариев**).
5. Проверьте синтаксис созданного сценария.
6. Для сохранения и контрольного запуска созданного сценария нажмите кнопку **Применить**.
4. В зависимости от режима работы Модуля нажмите кнопку **Применить** или кнопку **ОК** в окне настройки.

Примечание. Для выхода из окна редактирования без сохранения изменений нажмите кнопку **Отмена**.

2.2 Работа с Модулем

При работе Модуля для каждого скрипта создается отдельный исполнитель `nodejs.exe`, который запускает внешнее приложение `node.exe`. Логи исполнения `nodejs.exe` пишутся в стандартную для SecurOS директорию `%ProgramData%/logs/nodejs.N.log`.

В разделе рассмотрены следующие вопросы:

- **Создание и выполнение сценариев**;
- **Проверка синтаксиса и отладка сценария**;
- **Пакет securos**;
- **Установка и использование дополнительных пакетов**.

2.2.1 Создание и выполнение сценариев

Для использования скриптов необходимо:

1. Создать групповой объект *Скрипты Node.js* (группа *Интеграция и автоматизация*). Подробнее смотри раздел **Создание и настройка объекта Скрипты Node.js**;
2. Создать один или более дочерних объектов *Скрипт Node.js* (см. **Создание и настройка объекта Скрипт Node.js**). Назовите их предполагаемыми именами сценариев.
3. Наберите текст сценария в окне Модуля.
4. В зависимости от режима работы Модуля (отладка/исполнение скрипта) задайте прочие параметры скрипта (см. **Создание и настройка объекта Скрипт Node.js**).
5. Для выполнения сценария нажмите в окне Модуля кнопку **Применить**.

Примечание. Проверка синтаксиса сценария будет выполнена автоматически при выполнении скрипта. В режиме редактирования проверка не выполняется.

Для каждого объекта *Скрипт Node.js* запускается свой экземпляр исполнителя `nodejs(.exe)`. Каждый экземпляр исполнителя в свою очередь запускает свой экземпляр внешнего приложение `node(.exe)` как дочерний процесс, передавая ему пользовательский скрипт. Таким образом, скрипты запускаются и работают независимо друг от друга. Особенности взаимодействия исполнителя и приложения `node`:

- потоки `stdout/stderr` из процесса `node` выводятся в окно **Консоль** настроечной панели объекта *Скрипт Node.js*, а также параллельно пишутся в отдельный лог-файл `nodejs.[ID].console.log`;
- при изменении настроек объекта `Скрипт Node.js` исполнитель перезапускает приложение `node`;
- если исполнение скрипта запрещено в настройках объекта *Скрипт Node.js* — приложение приложение `node` будет остановлено;
- при остановке самого исполнителя (в случае отключения или удаления объекта *Скрипт Node.js* в *Дереве объектов*), приложение `node` так же будет остановлено.

2.2.2 Проверка синтаксиса и отладка сценария

В разделе рассмотрены следующие вопросы:

- **Проверка синтаксиса**;
- **Настройки подключения отладчика**;
- **Использование встроенного отладчика в браузерах Chrome и Edge**.

Проверка синтаксиса
Проверка синтаксиса выполняется при выполнении сценария. Сообщения об ошибках будут выведены в **Консоли** объекта *Скрипт Node.js* (см. **Настройка и интерфейс Модуля**).

Настройки подключения отладчика
Для использования отладчика необходимо:

- Разрешить подключение отладчиком в настройках объекта *Скрипт Node.js*;
- Выбрать в настройках объекта объекта *Скрипт Node.js* сетевой интерфейс для подключения: локальный или внешний.

При выборе локального сетевого интерфейса отладчик должен быть запущен на том же компьютере, на котором работает объект *Скрипт Node.js* (может иметь смысл в целях безопасности). При выборе внешнего сетевого интерфейса подключаться отладчиком можно с любого компьютера.

После запуска скрипта в **Консоль** будет выведено предупреждающее сообщение вида `Debugger listening on ws://172.16.1.133:9229/...` . Выведенный IP-адрес и порт используются при настройке подключения на стороне отладчика. При использовании отладчика должен использоваться режим `attaching` (подключение к работающему приложению).

Использование встроенного отладчика в браузерах Chrome и Edge
Браузеры Chrome и Edge имеют встроенный отладчик для приложений Node.js, поэтому являются наиболее доступными инструментами. Чтобы запустить режим отладки:

- Введите в адресной строке браузера служебный `url`:
  – `edge://inspect` — в браузере Edge;

– `chrome://inspect` — в браузере Chrome.

- На открывшейся странице перейдите по ссылке **Open dedicated DevTools for Node**, чтобы открыть окно отладчика.

Отладчик, как правило, автоматически подключается к локально работающему приложению `node`. Однако, в случае удаленного подключения (если в настройках объекта *Скрипт Node.js* был выбран внешний сетевой интерфейс), IP-адрес и порт необходимо ввести вручную. Для этого:

- Перейдите на вкладку **Connection**;
- Нажмите на кнопку **Add connection**;
- В появившемся поле введите **IP-адрес** и **Номер порта** (например, 172.16.1.133:9229) и нажмите на кнопку **Add**.

Примечание. IP-адрес и порт подключения выводятся в **Консоли** объекта *Скрипт Node.js* при запуске скрипта.

2.2.3 Пакет securos

Пакет `securos` поставляется в комплекте Модуля и содержит методы работы с ядром SecurOS. Взаимодействие с ядром осуществляется через исполнитель. Использование пакета отдельно от процесса исполнителя невозможно.

Для использования методов необходимо импортировать установленный пакет и вызвать функцию `connect`:

```
const securos = require('securos');
securos.connect(function(core) {
    // ... ваш код взаимодействия с ядром
});
```

Функция `connect` выполнит подключение к сервисам ядра и через callback-функцию вернёт объект `ICore`. Если внутри callback-функции вы хотите использовать оператор `await` — объявите её асинхронной:

```
const securos = require('securos');
securos.connect(async function(core) {
    // ... ваш код взаимодействия с ядром, в том числе операторы await
});
```

Также пакет включает в себя некоторые вспомогательные константы:

- объект `ObjectEvent` — содержит (в виде свойств) коды событий изменения конфигурации объекта (`ObjectEvent.CREATED`, `ObjectEvent.UPDATED` и `ObjectEvent.DELETED`);
- объект `MediaClientKey` — содержит (в виде свойств) коды клавиш для команды `MEDIA_CLIENT||KEY_PRESSED` (`MediaClientKey.UP`, `MediaClientKey.DOWN`, `MediaClientKey.REC` и т.д.).

2.2.3.1 Интерфейс ICore

В разделе описаны **Свойства** и **Методы** интерфейса `ICore`.

2.2.3.1.1 Свойства

В разделе описаны следующие свойства интерфейса `ICore`:

- **selfId**;
- **selfType**.

2.2.3.1.1.1 selfId

Свойство: `selfId: string`
Описание: ID текущего объекта *Скрипт Node.js*

2.2.3.1.1.2 selfType

Свойство: `selfType: string`
Описание: Тип объекта *Скрипт Node.js* (строка `NODEJS_SCRIPT`)

2.2.3.1.2 Методы

В разделе описаны следующие методы интерфейса `ICore`:

- **getObjectsIds**;
- **getObjectChildsIds**;
- **getObjectParentId**;
- **getObject**;
- **registerEventHandler**;
- **registerReact**;
- **registerObjectHandler**;
- **sendEvent**;
- **doReact**;
- **disconnect**.

2.2.3.1.2.1 getObjectsIds

Синтаксис: `getObjectsIds(type: string): Promise<string[]>`	
Описание: Получение списка ID всех объектов указанного типа	
Параметры:	
`type`	Идентификатор типа объекта SecurOS

Примечание. Метод представляет собой асинхронную операцию. При вызове возвращает новый объект `Promise`. Получить результат асинхронной операции можно с помощью метода `Promise.then()` или с помощью оператора `await`.

2.2.3.1.2.2 getObjectChildsIds

Синтаксис: `getObjectChildsIds(type: string, id: string, childType: string): Promise<string[]>`	
Описание: Получение массива ID всех объектов, дочерних объекту указанного типа	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор объекта SecurOS
`childType`	Идентификатор типа дочернего объекта SecurOS

Примечание. Метод представляет собой асинхронную операцию. При вызове возвращает новый объект `Promise`. Получить результат асинхронной операции можно с помощью метода `Promise.then()` или с помощью оператора `await`.

2.2.3.1.2.3 getObjectParentId

Синтаксис: `getObjectParentId(type: string, id: string): Promise<string>`	
Описание: Получение ID родительского объекта	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор объекта SecurOS

Синтаксис: `getObjectParentId(type: string, id: string, parentType: string): Promise<string|null>`	
Описание: Получение ID ближайшего родительского объекта указанного типа (если объект не найден, возвращает значение `NULL`)	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор объекта SecurOS
`parentType`	Тип родительского объекта SecurOS

Примечание. Метод представляет собой асинхронную операцию. При вызове возвращает новый объект `Promise`. Получить результат асинхронной операции можно с помощью метода `Promise.then()` или с помощью оператора `await`.

2.2.3.1.2.4 getObject

Синтаксис: `getObject(type: string, id: string): Promise<ICoreObject|null>`	
Описание: Получение копии конфигурации `ICoreObject` (см. **Интерфейс ICoreObject**) отдельного объекта (если объект не найден, возвращает значение `NULL`)	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор объекта SecurOS

Примечание. Метод представляет собой асинхронную операцию. При вызове возвращает новый объект `Promise`. Получить результат асинхронной операции можно с помощью метода `Promise.then()` или с помощью оператора `await`.

2.2.3.1.2.5 registerEventHandler

Синтаксис: `registerEventHandler(sourceType: string, sourceId: string, action: string, callback: (ev:IEventMsg)=>void) : ISubscription`	
Описание: Регистрирует функцию-обработчик, вызываемую при наступлении заданного события (`event`) заданного объекта	
Параметры:	
`sourceType`	Идентификатор типа объекта SecurOS
`sourceId`	Идентификатор объекта SecurOS типа `type`
`action`	Идентификатор события объекта SecurOS типа `type`
Возвращаемое значение:	
	Объект **ISubscription**. Чтобы иметь возможность удалить обработчик в будущем, сохраните ссылку на этот объект.

Примечание. При каждом вызове обработчику будет передаваться объект **IEventMsg**.

2.2.3.1.2.6 registerReact

Синтаксис: `registerReact(action: string, callback: (ev:IEventMsg)=>void):`

ISubscription	
Описание: Регистрирует функцию-обработчик, вызываемую при выполнении заданной команды (react) заданного объекта	
Параметры:	
action	Имя действия
Возвращаемое значение:	
	Объект **ISubscription**. Чтобы иметь возможность удалить обработчик в будущем, сохраните ссылку на этот объект.

Примечание. При каждом вызове обработчику будет передаваться объект **IEventMsg**.

2.2.3.1.2.7 registerObjectHandler

Синтаксис: registerObjectHandler(objectType: string, callback: (ev:IObjectEventMsg)=>void): ISubscription	
Описание: Регистрирует функцию-обработчик изменения конфигурации объектов заданного типа	
Параметры:	
objectType	Тип объекта SecurOS
Возвращаемое значение:	
	Объект **ISubscription**. Чтобы иметь возможность удалить обработчик в будущем, сохраните ссылку на этот объект.

Примечание. При каждом вызове обработчику будет передаваться объект **IObjectEventMsg**.

2.2.3.1.2.8 sendEvent

Синтаксис: sendEvent(sourceType: string, sourceId: string, action: string, params: object): void	
Описание: Отправляет событие с указанными значениями параметров от имени заданного объекта в систему SecurOS.	
Параметры:	
sourceType	Идентификатор типа объекта SecurOS
sourceId	Идентификатор объекта SecurOS типа sourceType
action	Идентификатор события объекта SecurOS типа sourceType

`params`	Имена и значения параметров события `event` (число параметров не ограничено)

2.2.3.1.2.9 doReact

Метод может быть использован для отправки команд как *объектам* (см. **Приложение 1. События и действия объектов SecurOS**), так и *сервисам* SecurOS (см. **Приложение 2. События и действия сервисов SecurOS**).

Различия в синтаксисе метода описаны в разделах ниже:

- **Отправка команд объекту SecurOS**.
- **Отправка команд сервису SecurOS**.

Отправка команд объекту SecurOS

При отправке команд *объекту* SecurOS метод используется со следующими параметрами (см. Таблицу **2**):

Таблица 2. Параметры метода при отправке команды объекту SecurOS

Синтаксис: `doReact(objectType: string, objectId: string, action: string, params: object): void`	
Описание: Отправляет команду SecurOS с указанными значениями параметров заданному объекту SecurOS.	
Параметры:	
`objectType`	Идентификатор типа объекта SecurOS
`objectId`	Идентификатор объекта SecurOS типа `objectType`
`action`	Идентификатор команды, отправляемой заданному объекту SecurOS типа `objectType`
`params`	Имена и значения параметров команды `react` (число параметров не ограничено)

Отправка команд сервису SecurOS

При отправке команд *сервису* SecurOS метод используется со следующими параметрами (см. Таблицу **3**):

Таблица 3. Параметры метода при отправке команды сервису SecurOS

Синтаксис: `doReact(serviceType: string, computerName: string, action: string, params: object): void`	
Описание: Отправляет команду SecurOS с указанными значениями параметров заданному сервису SecurOS.	
Параметры:	
`serviceType`	Идентификатор типа сервиса SecurOS

computerName	Имя компьютера, на котором запущен данный сервис SecurOS
action	Идентификатор команды, отправляемой заданному объекту SecurOS типа objectType
params	Имена и значения параметров команды react (число параметров не ограничено)

2.2.3.1.2.10 disconnect

Синтаксис: disconnect(): void
Описание: Удаление всех обработчиков и отключение от сервисов ядра

Метод вызывает отключение от ядра, после чего объект Core становится недействительным. Если больше нет никаких активных операций в цикле событий Node.js (запущенных таймеров, открытых сокетов и т.п.), то это также вызовет завершение работы скрипта (завершение приложения node).

2.2.3.2 Интерфейс ISubscription

На каждый обработчик, зарегистрированный методами **ICore.registerEventHandler**, **ICore.registerReact** или **ICore.registerObjectHandler**, создаётся и возвращается новый объект ISubscription. Если в будущем обработчик потребуется удалить, то объект ISubscription необходимо сохранить, а в нужное время для удаления обработчика — вызвать метод ISubscription.unregister().

2.2.3.2.1 Методы

В разделе описаны следующие методы интерфейса ISubscription:

- **unregister**.

2.2.3.2.1.1 unregister

Синтаксис: unregister(): void
Описание: Удаление обработчика

2.2.3.3 Интерфейс IEventMsg

Объект с интерфейсом IEventMsg — аргумент, с которым вызывается обработчик, зарегистрированный методами **ICore.registerEventHandler** или **ICore.registerReact**.

2.2.3.3.1 Свойства

В разделе описаны следующие свойства интерфейса IEventMsg:

- **sourceType**;
- **sourceId**;
- **action**;

- params.

2.2.3.3.1.1 sourceType

Свойство: `sourceType: string`

Описание: Тип объекта-отправителя события (`event`) или получателя команды (`react`)

2.2.3.3.1.2 sourceId

Свойство: `sourceId: string`

Описание: Идентификатор объекта-отправителя события (`event`) или получателя команды (`react`)

2.2.3.3.1.3 action

Свойство: `action: string`

Описание: Идентификатор события/команды

2.2.3.3.1.4 params

Свойство: `params: object`

Описание: Опциональные параметры события или команды (конкретный набор свойств зависит от типа события/команды)

2.2.3.4 Интерфейс IObjectEventMsg

Объект с интерфейсом `IObjectEventMsg` – аргумент, с которым вызывается обработчик, зарегистрированный методом **ICore.registerObjectHandler**.

2.2.3.4.1 Свойства

В разделе описаны следующие свойства интерфейса `IObjectEventMsg`:

- **sourceType**;
- **sourceId**;
- **action**;
- **params**.

Названия событий изменения конфигурации дерева объектов для упрощения написания кода также содержатся в пакете `securos` в виде следующих констант:

- `securos.ObjectEvent.CREATED`;
- `securos.ObjectEvent.UPDATED`;
- `securos.ObjectEvent.DELETED`.

2.2.3.4.1.1 sourceType

Свойство: `sourceType: string`

Описание: Тип объекта с изменившейся конфигурацией

2.2.3.4.1.2 sourceId

Свойство: `sourceId: string`

Описание: ID объекта с изменившейся конфигурацией

2.2.3.4.1.3 action

Свойство: `action: string`

Описание: Название события изменения конфигурации: `CREATED` (объект создан), `CHANGED` (изменение параметров объекта) или `DELETED` (объект удален)

2.2.3.4.1.4 params

Свойство: `params: object`

Описание: Параметры конфигурации объекта: весь набор параметров (событие `CREATED`) или изменившиеся параметры (событие `UPDATED`)

2.2.3.5 Интерфейс ICoreObject

Объект с интерфейсом `ICoreObject` содержит копию конфигурации отдельного объекта в дереве объектов SecurOS. Получить копию данных можно с помощью метода **ICore.getObject**. **Свойства** и **Методы** интерфейса `ICoreObject` описаны ниже.

2.2.3.5.1 Свойства

В разделе описаны следующие свойства интерфейса `ICoreObject`:

- **enabled**;
- **type**;
- **id**;
- **name**;
- **parentType**;
- **parentId**;
- **childTypes**;
- **state**;
- **params**.

2.2.3.5.1.1 enabled

Свойство: `enabled: boolean`

Описание: Состояние объекта (включен/отключен)

2.2.3.5.1.2 type

Свойство: `type: string`

Описание: Тип объекта

2.2.3.5.1.3 id

Свойство: `id: string`

Описание: ID объекта

2.2.3.5.1.4 name

Свойство: `name: string`

Описание: Имя объекта в конфигурации

2.2.3.5.1.5 parentType

Свойство: `parentType: string`

Описание: Тип родительского объекта

2.2.3.5.1.6 parentId

Свойство: `parentId: string|null`

Описание: ID родительского объекта (если родителя нет, принимает значение `NULL`)

2.2.3.5.1.7 childTypes

Свойство: `childTypes: string[]`

Описание: Список возможных типов объектов, которые могут быть дочерними текущему объекту

2.2.3.5.1.8 state

Свойство: `state: string`
Описание: Текущее состояние объекта (копия параметра `state`). Набор возможных значений зависит от типа объекта

2.2.3.5.1.9 params

Свойство: `params: Object`
Описание: Все настройки и текущие значения параметров объекта

2.2.3.5.2 Методы

В разделе описаны следующие методы интерфейса `ICoreObject`:

- **getChildsIds**;
- **getParentId**.

Данные методы — асинхронные операции. Каждый из методов при вызове возвращает новый объект `Promise`. Результат асинхронной операции можно получить с помощью метода `Promise.then()` или с помощью оператора `await` (см. **Promise API**, **Асинхронные функции**).

Методы являются вспомогательными. Вместо них можно использовать методы **ICore.getObjectChildsIds()** и **ICore.getObjectParentId()**.

2.2.3.5.2.1 getChildsIds

Синтаксис: `getChildsIds(childType: string): Promise<string[]>`	
Описание: Получение списка ID дочерних объектов указанного типа	
Параметры:	
`childType`	Идентификатор типа дочернего объекта SecurOS

2.2.3.5.2.2 getParentId

Синтаксис: `getParentId(parentType: string): Promise<string|null>`	
Описание: Получение ID ближайшего родительского объекта указанного типа	
Параметры:	
`parentType`	Идентификатор типа родительского объекта SecurOS

2.2.3.6 Примеры сценариев

Ниже приведены примеры сценариев с использованием методов пакета `securos`:

- **Подключение к ядру SecurOS**;
- **Получение списка Устройств видеозахвата и Камер в дереве SecurOS**;
- **Оформление подписки на событие**;
- **Отправка события в ядро SecurOS**.

Подключение к ядру SecurOS

В данном примере скрипт:

1. Подключается к ядру;
2. Выводит в консоль свой тип объекта и ID;
3. Отключается от ядра и завершает работу.

Листинг 2. Подключение к ядру SecurOS

```
const securos = require('securos');
securos.connect(function (core) {
    console.log(`Тип объекта ${core.selfType}',
                ID объекта [${core.selfId}]`);
    core.disconnect();
});
```

Пример вывода в консоль объекта *Скрипт Node.js*:

Листинг 3. Вывод результата работы скрипта

```
2020-10-02T16:31:35.975: запуск 'node.js'
2020-10-02T16:31:35.978: 'node.js' запущен
2020-10-02T16:31:36.062: Тип объекта NODEJS_SCRIPT', ID объекта [1]

2020-10-02T16:31:36.072: 'node.js' выполнен успешно, код 0
```

Получение списка Устройств видеозахвата и Камер в дереве SecurOS

В данном примере скрипт подключается к ядру, получает список *Устройств видеозахвата*, их конфигурации и список *Камер*, и выводит всю эту информацию в консоль:

Листинг 4. Получение списка Устройств видеозахвата и Камер в дереве SecurOS

```
const securos = require('securos');
console.log('');

securos.connect(async function(core) {
    let grabbers = await core.getObjectsIds("GRABBER");
    grabbers = grabbers.sort();

    for (grabberId of grabbers) {
        console.log(`Граббер [${grabberId}]`);

        const grabber = await core.getObject('GRABBER', grabberId);
        console.log(` -- имя: '${grabber.name}'`);
        console.log(` -- состояние: ${grabber.enabled ? 'включен':
```

```
                'отключен'}`);
        console.log(` -- модель: '${grabber.params.model}'/
                '${grabber.params.type}'`);
        console.log(` -- IP: ${grabber.params.ip}`);
        console.log(` -- камеры:`);

        const cams = (await grabber.getChildsIds('CAM')).sort();
        for (const camId of cams) {
            const cam = await core.getObject('CAM', camId);
            console.log(`    -- камера '${cam.name}'[${cam.id}],
                    ${cam.enabled ? 'включена': 'отключена'},
                    ${cam.state}`);
        }
    }
});
```

Пример вывода в консоль объекта *Скрипт Node.js*:

Листинг 5. Вывод результата работы скрипта

```
2020-10-02T21:58:50.584: запуск 'node.js'
2020-10-02T21:58:50.586: 'node.js' запущен
2020-10-02T21:58:50.673:
Граббер [1]
 -- имя: 'Устройство видеозахвата 1'
 -- состояние: включен
 -- модель: 'default'/'Virtual'
 -- IP:
 -- камеры:
    -- камера 'Камера 1'[1], включена, DISARMED
    -- камера 'Камера 2'[2], отключена, DISARMED
Граббер [2]
 -- имя: 'AVI'
 -- состояние: включен
 -- модель: 'default'/'Player AVI'
 -- IP: E:\work\tasks\tk-kad\videos\240 (2020-09-14 14'01'02 -
        2020-09-14 14'08'38).avi
 -- камеры:
    -- камера 'Камера 3'[3], включена, DISARMED
Граббер [3]
 -- имя: 'Устройство видеозахвата 3'
 -- состояние: отключен
 -- модель: 'ISAPI (default)'/'Hikvision'
 -- IP: 172.16.16.11
 -- камеры:
    -- камера 'Камера 4'[4], отключена, DISARMED
```

Оформление подписки на событие

В данном примере скрипт:

1. Устанавливает обработчик на событие `CAM|1|REC` (камера с `ID [1]` поставлена на запись);
2. При каждом событии выводит в консоль сообщение `Камера поставлена на запись`;
3. После обработки третьего по счету события удаляет обработчик.

Листинг 6. Оформление подписки на событие

```
const securos = require('securos');
securos.connect(function(core) {
    console.log("");
    let eventsCount = 0;
    const handler = core.registerEventHandler("CAM", "1", "REC",
                                                function(msg) {
        console.log("Камера поставлена на запись");
        if (++eventsCount > 2) {
            handler.unregister();
            console.log("Обработчик удалён");
        }
    });
});
```

Отправка события в ядро SecurOS

В данном примере скрипт отправляет в ядро некий выдуманный тип сообщения со следующими параметрами:

1. `sourceType` — собственный тип (`NODEJS_SCRIPT`);
2. `sourceId` — собственный `ID`;
3. `action` — некое выдуманное название сообщения (`TEST_EVENT`);
4. `параметры` — некий произвольно выбранный набор параметров. Обратите внимание, что параметры передаются в виде JavaScript-объекта (набор `имя_свойства+значение`).

Листинг 7. Отправка события в ядро SecurOS

```
const securos = require('securos');
securos.connect(function(core) {
    // Вы можете создать параметры отдельно:
    let params = {
        comment: 'тест посылки события',
        param1: 1,
        param2: 2
    };
    core.sendEvent(core.selfType, core.selfId, "TEST_EVENT", params);

    // Вы также можете передать параметры в виде литерала объекта:
    // core.sendEvent(core.selfType, core.selfId, "TEST_EVENT",
    // {comment: 'тест посылки события', param1: 1, param2: 2});
});
```

2.2.4 Установка и использование дополнительных пакетов

Для установки дополнительных пакетов используется утилита `npm`.

> **Внимание!** В операционной системе Linux утилита `npm` устанавливается отдельно, через системный менеджер пакетов (например, `apt-get`). Чтобы проверить наличие утилиты `npm`, откройте командную строку и введите `npm --version`. Если окажется, что такой команды нет, утилиту потребуется установить (например, командой `sudo apt install npm`).

Для установки пакета необходимо:

- Открыть командную строку и перейти в рабочий каталог скриптов (`[node.js]`).

- Ввести команду `npm install <имя_пакета>`.

Пакет и все его зависимости (другие необходимые пакеты) будут установлены в каталог `[node_modules]` (зависимости могут установлены как в отдельных каталогах, так и внутри каталога с пакетом в подкаталоге `[node_modules]`).

Примечание. Отличия в структуре каталогов для платформ Windows и Linux смотри в разделах **Особенности реализации на Windows** и **Особенности реализации на Linux**.

В случае, если уже имеются все исходные файлы пакета со всеми его зависимостями, достаточно скопировать их в каталог `[node_modules]`. Обратите внимание: некоторые пакеты включают в себя не только JavaScript-модули, но и бинарные модули, которые компилируются из исходных кодов (при установке пакета через утилиту `npm` это делается автоматически). Поэтому такой способ может не работать при переносе пакета, ранее установленного для другой платформы/архитектуры или другой версии Node.js.

2.2.4.1 Создание собственных модулей

Возможно, что какие-то повторяющиеся в разных скриптах операции будет разумно оформить в виде функций в одном или нескольких модулей. Здесь может возникнуть два вопроса:

1. Где располагать свои собственные модули на диске?
2. Как оформить свои собственные функции в виде модулей?

При импорте модуля в качестве аргумента функции `require` должен указываться относительный путь. Путь указывается относительно роабочего каталога скриптов, а в случае импорта из другого модуля — путь относительно расположения этого модуля. Это единственное, что нужно принять во внимание при выборе места расположение модулей. Разумно размещать свои модули в отдельном каталоге.

Оформление самого модуля выполняется достаточно просто. Рассмотрим это на следующем примере:

1. Пользовательский модуль `my-module.js` расположен в каталоге `iss/securos/node.js/my-libs`;
2. Модуль экспортирует функцию `sum` (возвращающую результат сложения) и строковую константу `value`.

Листинг 8. Код модуля

```
// файл /iss/securos/node.js/my-libs/my-module.js
function sum(a, b) {
    return a + b;
}

module.exports = {
    sum: sum,
    value: '12345'
};
```

Листинг 9. Код скрипта, использующего модуль

```
const m = require('./my-libs/my-module.js');
```

```
m.sum(2, 3);      // результат вызова функции - число '5'
m.value;          // свойство содержит строку со значением '12345'
```

3 Разработка web-приложений

Система SecurOS позволяет создать удобный интерфейс пользователя посредством HTML-окон.

Для выполнения web-приложений в SecurOS предназначены объекты *HTML5 FrontEnd* и *Всплывающее окно HTML5* (подробнее см. **Руководство администратора SecurOS**). Особенности web-приложений SecurOS приведены в Таблице 4.

Внимание! Окна для выполнения web-приложений не являются полноценными браузерами и обладают ограниченной функциональностью. При использовании функций, не указанных в Таблице 4 (например, навигации по страницам, ручного обновления страниц, дополнительных всплывающих окон, Drag and Drop и др.), корректная работа web-приложений SecurOS не гарантируется.

Таблица 4. Особенности web-приложений SecurOS

Параметр	Описание
Работа с HTML-страницами	Загрузка единственной HTML-страницы и выполнение ее JavaScript кода. URL HTML-страницы указывается в параметрах объекта SecurOS
Выполнение CORS-запросов	Без ограничений
Скачивание файлов	Поддерживается
Декодирование видео формата H.264	Поддерживается
Очистка кэша при перезапуске процесса-исполнителя	Поддерживается

3.1 Объект ISScustomAPI

Объект предоставляет методы для создания пользовательских интерфейсов таких объектов SecurOS, как *HTML5 FrontEnd* и *Всплывающее окно HTML5* (см. **Руководство администратора SecurOS**):

- **getSelfId**;
- **hidePopup**;
- **showPopup**;
- **onEvent**;
- **onCommand**;
- **onSetup**;
- **sendEvent**;

- **sendReact**;
- **subscribe**;
- **unsubscribe**.

Команды для управления *Всплывающим окном HTML5* представлены в разделе **Всплывающее окно HTML5**.

Пример пользовательского интерфейса управления с применением методов *ISScustomAPI* представлен в разделе **Пример HTML5-окна**.

3.1.1 getSelfId

Синтаксис: `getSelfId ()`

Описание: Возвращает идентификатор пользовательского интерфейса, реализованного объектом SecurOS *HTML5 FrontEnd* или *Всплывающее окно HTML5*.

Параметры: нет

3.1.2 hidePopup

Синтаксис: `hidePopup ()`

Описание: Скрывает **Всплывающее окно HTML5**.

Параметры: нет

3.1.3 showPopup

Синтаксис: `showPopup (x, y, w, h, display)`

Описание: Отображает **Всплывающее окно HTML5** на мониторе того *Компьютера*, под которым был создан объект. Если окно создано в *Профиле рабочего места оператора*, то будет отображено на мониторах всех *Рабочих мест оператора*, которые используют данный профиль в качестве *Рабочего окружения*.

Параметры:

Примечание. Перечисленные ниже параметры являются опциональными, но в случае использования должны передаваться совместно. Если не заданы, будут использоваться значения `0,0,100,100,1`.

`x`	Отступ от левого края в процентах от ширины экрана
`y`	Отступ от верхнего края в процентах от высоты экрана
`w`	Ширина окна в процентах от ширины экрана
`h`	Высота окна в процентах от высоты экрана

`display`	Идентификатор физического монитора, на котором будет отображаться окно

3.1.4 onEvent

Внимание! Для использования функции необходимо предварительно подписаться на события с помощью функции **subscribe**.

Синтаксис: `onEvent (callback_function)`	
Описание: Добавление функции-обработчика события с указанными параметрами, сгенерированного указанным объектом. Функция-обработчик принимает список указанных ниже параметров, возвращаемых callback-функцией.	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор (номер) объекта SecurOS типа `type`
`action`	Идентификатор события объекта SecurOS типа `type`
`params`	JavaScript-объект со свойствами, одноименными с параметрами события (количество параметров неограниченно)

3.1.5 onCommand

Внимание! Для использования функции необходимо предварительно подписаться на события с помощью функции **subscribe**.

Синтаксис: `onCommand (callback_function)`	
Описание: Добавление функции-обработчика команды с указанными параметрами, отправленной указанному объекту. Функция-обработчик принимает список указанных ниже параметров, возвращаемых callback-функцией.	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор (номер) объекта SecurOS типа `type`
`action`	Идентификатор команды объекта SecurOS типа `type`
`params`	JavaScript-объект со свойствами, одноименными с параметрами команды (количество параметров неограниченно)

Допустимые комбинации параметров функции приведены ниже:

- `ISScustomAPI.onCommand((action, params) => {...}).`
- `ISScustomAPI.onCommand((type, id, action, params) => {...}).`

3.1.6 onSetup

Синтаксис: `onSetup (callback_function)`	
Описание: Добавление функции-обработчика события изменения настроек объекта *HTML5 FrontEnd* в SecurOS (событие `SETUP`). Функция-обработчик принимает список указанных ниже параметров, упакованных в формат JSON.	
Принимаемые параметры:	
`media_client_id`	Идентификатор объекта *Медиа Клиент*, выбранного в настройках объекта *HTML5 FrontEnd*
`map_window_id`	Идентификатор объекта *Карта: Интерфейс оператора*, выбранного в настройках объекта *HTML5 FrontEnd*
`advanced`	Набор дополнительных параметров, заданных в настройках объекта *HTML5 FrontEnd* (поле **Дополнительно**). Дополнительные параметры передаются в том формате, в каком они были заданы в настройках объекта.
`operator`	Логин оператора SecurOS, который использовался для запуска клиента SecurOS.

3.1.7 sendEvent

Синтаксис: `sendEvent (type, id, action, params)`	
Описание: Отправляет событие с указанными значениями параметров от имени заданного объекта в систему SecurOS.	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор (номер) объекта SecurOS типа `type`
`action`	Идентификатор события объекта SecurOS типа `type`
`params`	Набор параметров события, упакованный в формат JSON (количество параметров неограниченно)

3.1.8 sendReact

Синтаксис: `sendReact (type, id, action, params)`
Описание: Отправляет команду SecurOS с указанными значениями параметров от имени заданного объекта в систему SecurOS.
Параметры:

`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор (номер) объекта SecurOS типа `type`
`action`	Имя команды, отправляемой объекту SecurOS типа `type`
`params`	Набор параметров команды, упакованный в формат JSON (количество параметров неограниченно)

3.1.9 subscribe

Синтаксис: `subscribe (type, id, action)`	
Описание: Функция подписки на событие с заданными параметрами.	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор (номер) объекта SecurOS типа `type`
`action`	Идентификатор события объекта SecurOS типа `type`

Примечание. Чтобы подписаться на все события всех объектов, присвойте параметрам `id` и `action` значение "*" (звездочка).

Внимание! Подписка, оформленная индивидуально на события объекта, является независимой.

3.1.10 unsubscribe

Синтаксис: `unsubscribe (type, id, action)`	
Описание: Функция отмены подписки на событие с заданными параметрами.	
Параметры:	
`type`	Идентификатор типа объекта SecurOS
`id`	Идентификатор (номер) объекта SecurOS типа `type`
`action`	Идентификатор события объекта SecurOS типа `type`

Примечание. Чтобы отменить подписку на все события, оформленную с помощью "*" (см. **subscribe**), присвойте параметрам `id` и `action` значение "*" (звездочка).

Внимание! Отмена подписки, оформленной индивидуально, выполняется также индивидуально.

3.2 Всплывающее окно HTML5

Идентификатор типа объекта: `HTML_POPUP`.

В HTML-документе, загруженном в данный объект SecurOS, могут быть использованы все методы *ISScustomAPI* (см. **Объект ISScustomAPI**).

События: отсутствуют.

Таблица 5. Действия `HTML_POPUP`

Идентификатор действия: `SHOW`

Внимание! Если используется *Вплывающее окно HTML5*, созданное в *Профиле рабочего места оператора*, действие выполнено не будет. В этом случае для отображения *Всплывающего окна HTML5* используйте метод **showPopup**.

Название в макрокоманде: нет

Описание: Отобразить окно

Примечание. Перечисленные ниже параметры являются опциональными, но в случае использования должны передаваться совместно.

Параметры:

`x`	Отступ от левого края в процентах от ширины экрана
`y`	Отступ от верхнего края в процентах от высоты экрана
`w`	Ширина окна в процентах от ширины экрана
`h`	Высота окна в процентах от высоты экрана
`display`	Идентификатор физического монитора, на котором будет отображаться окно

Идентификатор действия: `HIDE`

Внимание! Если используется *Вплывающее окно HTML5*, созданное в *Профиле рабочего места оператора*, действие выполнено не будет. В этом случае для скрытия *Всплывающего окна HTML5* используйте метод **hidePopup**.

Название в макрокоманде: нет

Описание: Скрыть окно

Параметры: нет

3.3 Пример HTML5-окна

Для интеграции и применения пользовательского интерфейса в SecurOS необходимо создать HTML-документ и настроить объект *HTML5 FrontEnd / Всплывающее окно HTML5*:

- **HTML-документ —** текстовый файл, содержащий HTML-код, с помощью которого реализуется внешняя форма и алгоритм управления;
- **HTML5 FrontEnd / Всплывающее окно HTML5 —** объекты SecurOS (см. **Руководство администратора SecurOS**), с помощью которых пользовательский интерфейс интегрируется в SecurOS.

Ниже приведен пример пользовательского интерфейса управления SecurOS.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <meta name="generator" content="HTML Tidy for Windows (vers 15 August 2007), see
www.w3.org">
    <meta http-equiv="Content-Type" content="text/html; charset=us-ascii">
    <script type="text/javascript">
                        window.onload = function() {
                                document.getElementById("IssMessageType").value = 2;

                                ISScustomAPI.onSetup(function(settings){
                                        document.getElementById("settings").innerHTML =
settings;
                                        let jsonSettings = JSON.parse(settings);
                                        document.getElementById("MediaClientId").value =
jsonSettings.media_client_id;
                                        document.getElementById("ID").value =
jsonSettings.media_client_id;
                                        document.getElementById("MapWindowId").value =
jsonSettings.map_window_id;
                                });
                                ISScustomAPI.onEvent(function(type, id, action, params){
                                        var textArea = document.getElementById("events");
                                        textArea.value = type + '|' + id + '|' + action +
'|' + JSON.stringify(params) + '\n' + textArea.value;
                                });
                                ISScustomAPI.onCommand(function(action, params){
                                        var textArea = document.getElementById("reacts");
                                        textArea.value = action + '|' +
JSON.stringify(params) + '\n' + textArea.value;
                                });
                        }
    </script>
    <script language="JavaScript" type="text/javascript">
                        function MakeLiveAddSequenceParams()
                        {
                                var cameras = document.getElementById("Cameras").value;
                                var params = {
                                        seq : cameras.split(',').join('|')
                                };
                                return JSON.stringify(params);
                        }
                        function CamId()
                        {
                                var cameras = document.getElementById("Cameras").value;
                                cameras = cameras.split(',').join('|');
                                return cameras.split('|')[0];
```

```
                        }
                        function SendMessage()
                        {
                                var issMsgType =
document.getElementById("IssMessageType").value;
                                var action = document.getElementById("Action").value;
                                var type = document.getElementById("Type").value;
                                var id = document.getElementById("ID").value;
                                var params = document.getElementById("Params").value;
                                if (issMsgType === "1")
                                        ISScustomAPI.sendEvent(type, id, action, params);
                                else
                                        ISScustomAPI.sendReact(type, id, action, params);
                        }
                        function MakeFindLevelParams()
                        {
                                var params = {
                                        name : document.getElementById("LevelName").value
                                };
                                return JSON.stringify(params);
                        }
                        function MakeFindObjectParams()
                        {
                                var params = {
                                        type : document.getElementById("ObjectType").value,
                                        id : document.getElementById("ObjectId").value
                                };
                                return JSON.stringify(params);
                        }
                        function Subscribe(subscription)
                        {
                                var type =
document.getElementById("SubscriptionType").value;
                                var id = document.getElementById("SubscriptionID").value;
                                var action =
document.getElementById("SubscriptionAction").value;
                                if (subscription)
                                        ISScustomAPI.subscribe(type, id, action);
                                else
                                        ISScustomAPI.unsubscribe(type, id, action);
                        }
    </script>
    <style type="text/css">
        html {
            height: 100%;
            width: 100%;
        }

        #events {
            width: 100%;
        }

        #reacts {
            width: 100%;
        }
    </style>
    <title></title>
</head>
<body>
    <img src="https://issivs.com/wp-content/uploads/2019/10/1NEW-iss-logo3.png"
alt="logo"><br>
    <textarea id="settings"></textarea>
    <br>
```

```
    <br>
    <br>
    <br>
    <hr>
    ISS Message Type: <select id="IssMessageType">
        <option value="1">Event</option>
        <option value="2">React</option>
    </select> Action:
    <form>
        <input id="Action" value="SEEK"><br>
        <br>
        Type: <input id="Type" value="MEDIA_CLIENT"> ID: <input id="ID" value="1"><br>
        <br>
        Params: <input id="Params" value='{"date":"18-09-
19","time":"18:09:00:000","cam":"1"}'><br>
        <br>
        <input id="SendMessageBtn" value="Send Message" type="button"
onclick="SendMessage()"><br>
        <br>
    </form>
    <hr>
    Media Client ID:
    <form>
        <input id="MediaClientId" type="text" value="1"> Camera IDs: <input id="Cameras"
type="text" value="1,2,3" title="Use ',' or '|' as a separator"><br>
        <br>
        Send React example: <input id="SendAddSequence" value="Show live cameras"
type="button"
onclick="ISScustomAPI.sendReact('MEDIA_CLIENT',document.getElementById('MediaClientId').valu
e,'ADD_SEQUENCE',MakeLiveAddSequenceParams())"><br>
        <br>
        Send Event example: <input id="SendMdStart" value="Alarm camera" type="button"
onclick="ISScustomAPI.sendEvent('CAM',CamId(),'MD_START','')"><br>
        <br>
    </form>
    <hr>
    Map Window ID:
    <form>
        <input id="MapWindowId" type="text" value="1"><br>
        <br>
        Level name: <input id="LevelName" type="text" value="Level 1"> <input id="FindLevel"
 value="Find level" type="button"
onclick="ISScustomAPI.sendReact('MAP3',document.getElementById('MapWindowId').value,'FIND_LE
VEL',MakeFindLevelParams())"><br>
        <br>
        Object Type: <input id="ObjectType" type="text" value="CAM"> Object ID: <input
id="ObjectId" type="text" value="1"> <input id="FindObject" value="Find object"
type="button"
onclick="ISScustomAPI.sendReact('MAP3',document.getElementById('MapWindowId').value,'FIND_OB
JECT',MakeFindObjectParams())"><br>
        <br>
    </form>
    <hr>
    Type:
    <form>
        <input id="SubscriptionType" value="CAM"> ID: <input id="SubscriptionID" value="*">
Action: <input id="SubscriptionAction" value="*"><br>
        <br>
        <input id="SubscribeBtn" value="Subscribe" type="button" onclick="Subscribe(true)">
<input id="UnsubscribeBtn" value="Unsubscribe" type="button" onclick="Subscribe(false)"><br>
        <br>
        Received events:<br>
        <textarea id="events" rows="10"></textarea>
```

```
        <br>
        <br>
        Received commands:<br>
        <textarea id="reacts" rows="10"></textarea>
        <br>
        <br>
    </form>
    <hr>
    <form method="get" action="data.bin"><button type="submit">Download!</button></form>
    <a href="data.bin" download="data.bin">Download</a>
</body>
</html>
```

3.3.1 Отладка HTML5-окна

В разделе приводится описание процедуры удаленной отладки внешнего приложения с помощью браузера Google Chrome:

- **Для разработчиков**.
- **Для конечного пользователя**.

Для разработчиков

Отладка и профилирование внешних приложений выполняется с помощью Qt WebEngine. Ниже рассматриваются следующие вопросы:

- **Вывод информации на консоль**.
- **Инструменты разработчика Qt WebEngine**.
- **Использование аргументов командной строки**.
- **Очистка кэша**.

Вывод информации на консоль

JavaScript, выполняемый внутри Qt WebEngine, может использовать API консоли Chrome для вывода отладочной информации на консоль. Сообщения с отладочной информацией перенаправляются объектам логирования Qt внутри категории логирования `js`. По умолчанию выводятся только предупреждения и сообщения о фатальных ошибках. Чтобы изменить уровень логирования, вы должны либо установить пользовательские правила для категории `js`, либо предоставить пользовательские обработчики сообщений, переопределив `QWebEnginePage::javaScriptConsoleMessage()`, или подключившись к `WebEngineView::javaScriptConsoleMessage()`.

Все сообщения также можно просматривать с помощью инструментов разработчика Qt WebEngine.

Инструменты разработчика Qt WebEngine

Модуль Qt WebEngine предоставляет инструменты веб-разработчика, которые позволяют легко проверять и устранять проблемы макета и производительности любого веб-содержимого.

Доступ к инструментам разработчика осуществляется на локальной веб-странице с помощью браузера на основе Chromium или Qt WebEngine, например браузера Chrome. Чтобы активировать инструменты разработчика, запустите приложение, которое использует Qt WebEngine, со следующими аргументами командной строки:

`--remote-debugging-port=<port_number>`, где

<port_number> — локальный сетевой порт. Чтобы получить доступ к инструментам веб-разработчика, задайте в браузере адрес http://localhost:<port_number>.

В качестве альтернативы можно установить переменную окружения QTWEBENGINE_REMOTE_DEBUGGING. Значение переменной может представлять собой либо просто номер порта (аналогично аргументу --remote-debugging-port), либо адрес хоста и номер порта. Последнее значение может использоваться для указания сетевого интерфейса, который будет использоваться для экспорта, что позволит получить доступ к инструментам разработчика с удаленного устройства.

Подробное описание инструментов разработчика приведено в статье **Chrome DevTools**.

Использование аргументов командной строки

Для получения информации для отчетов об ошибках, при отладке можно использовать следующие аргументы командной строки:

- --disable-gpu — отключает аппаратное ускорение графического процессора. Это полезно при диагностике проблем OpenGL.
- --disable-logging — отключает ведение журнала консоли, что может быть полезно для отладочных сборок.
- --enable-logging --log-level=0 — включает ведение журнала консоли и устанавливает уровень ведения журнала равным 0, что означает, что в журнал выводятся сообщения с уровнем важности info и выше. Это значение по умолчанию для отладочных сборок. Другие возможные уровни журнала:
  - 1 — для предупреждений;
  - 2 — для ошибок;
  - 3 — для фатальных ошибок.
- --v=1 — увеличивает уровень ведения журнала по сравнению с --log-level и включает запись отладочных сообщений до уровня детализации 1. Более высокое число еще больше увеличивает детализацию, но может привести к большому количеству зарегистрированных сообщений. По умолчанию 0 (нет сообщений отладки).
- --no-sandbox — отключает песочницу для процессов рендерера и плагина. Имейте в виду, что отключение песочницы может представлять угрозу безопасности.
- --single-process — запускает рендерер и плагины в том же процессе, что и браузер. Это полезно для получения трассировки стека при сбоях рендерера.

В качестве альтернативы можно установить переменную окружения QTWEBENGINE_CHROMIUM_FLAGS. Например, следующее значение может быть установлено, чтобы отключить ведение журнала при отладке приложения с именем *mybrowser*:

```
QTWEBENGINE_CHROMIUM_FLAGS="--disable-logging" mybrowser
```

Очистка кэша

Содержимое формы, в том числе скрипты, кэшируется. Это может приводить к отображению устаревших данных после изменения исходного кода. Чтобы очистить кэш, выключите и снова включите соответствующий объект *HTML5 FrontEnd* в дереве объектов SecurOS (см. **Руководство администратора SecurOS**).

Для конечного пользователя

Для настройки режима отладки выполните следующие действия:

1. Откройте диалог **Выполнить**, для чего нажмите комбинацию клавиш **Win+R**.
2. В диалоговом окне введите строку `rundll32.exe sysdm.cpl, EditEnvironmentVariables` (см. рис. 2).

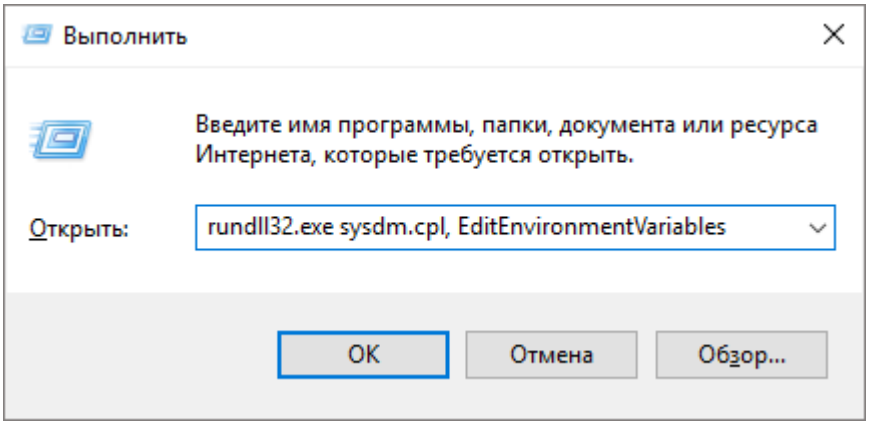

Рис. 2. Диалог Выполнить

Примечание. Для возможности создания системных переменных выполните команду от имени администратора, для чего нажмите комбинацию клавиш **Ctrl+Shift+Enter**.

---

3. В окне **Переменные среды** в блоке **Системные переменные** нажмите кнопку **Создать** (см. рис. 3).

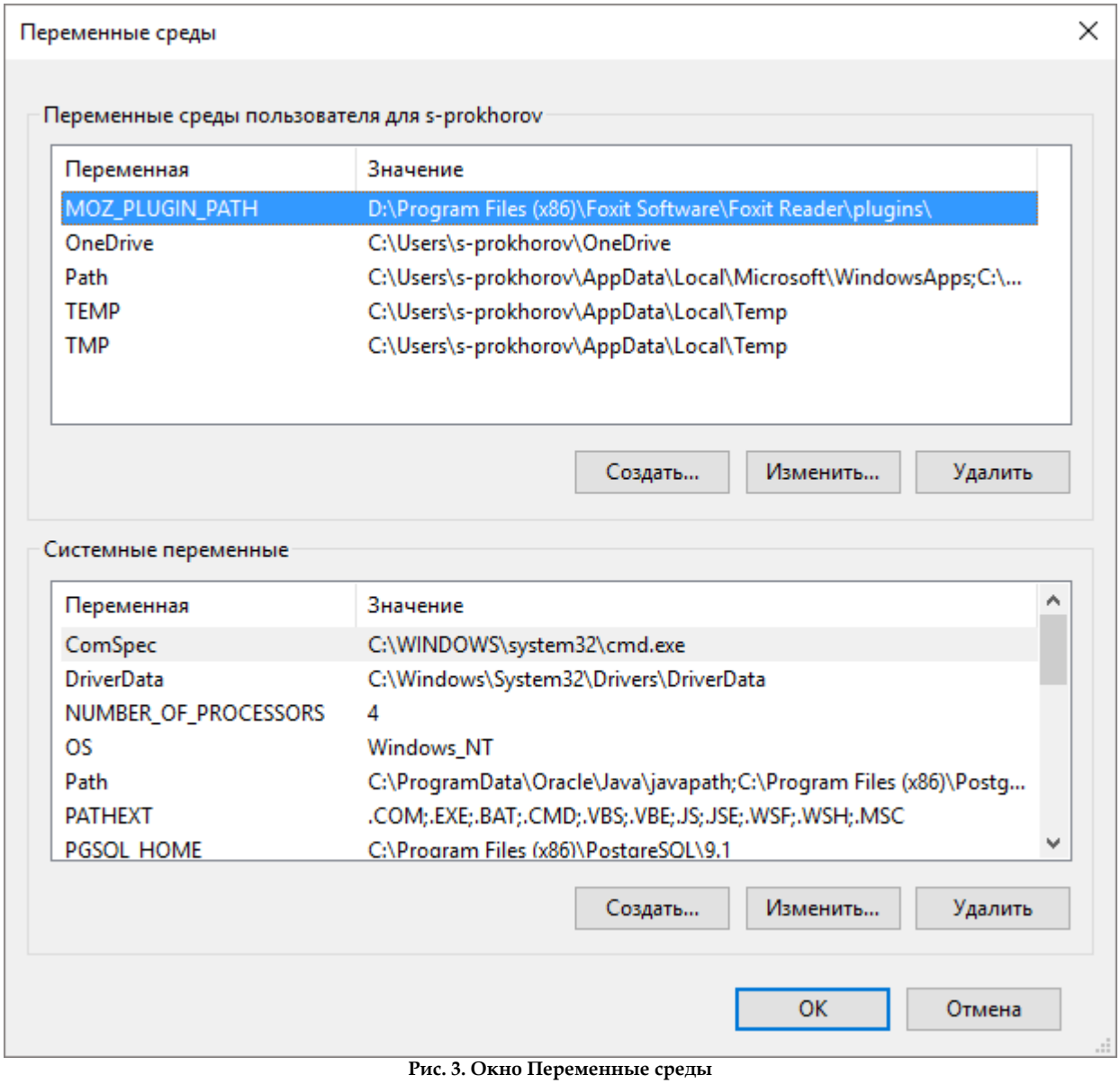

Рис. 3. Окно Переменные среды

4. В окне **Новая системная переменная** (см. рис. 4) задайте имя переменной (`QTWEBENGINE_REMOTE_DEBUGGING`) и номер порта для подключения отладчика (`44332`).

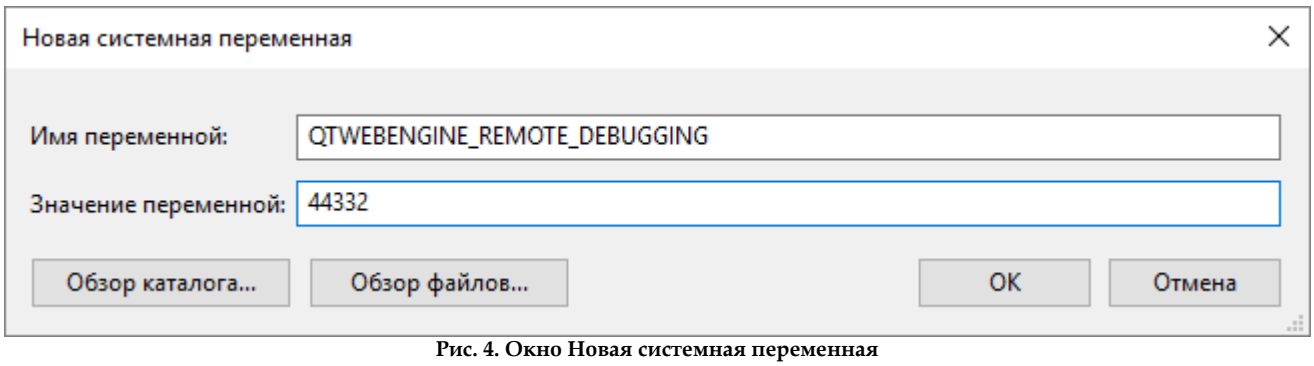

Рис. 4. Окно Новая системная переменная

5. Перезапустите службу *SecurOS Control Service*.
6. Запустите SecurOS.
7. Установите на компьютере браузер Google Chrome и запустите его.
8. В адресной строке браузера введите `http://127.0.0.1:44332`.

9. После подключения нажмите клавишу **F12**.
10. Браузер отобразит окно отладки web-приложения, выбранного в настройках объекта *HTML5 FrontEnd* в SecurOS (см. рис. 5).

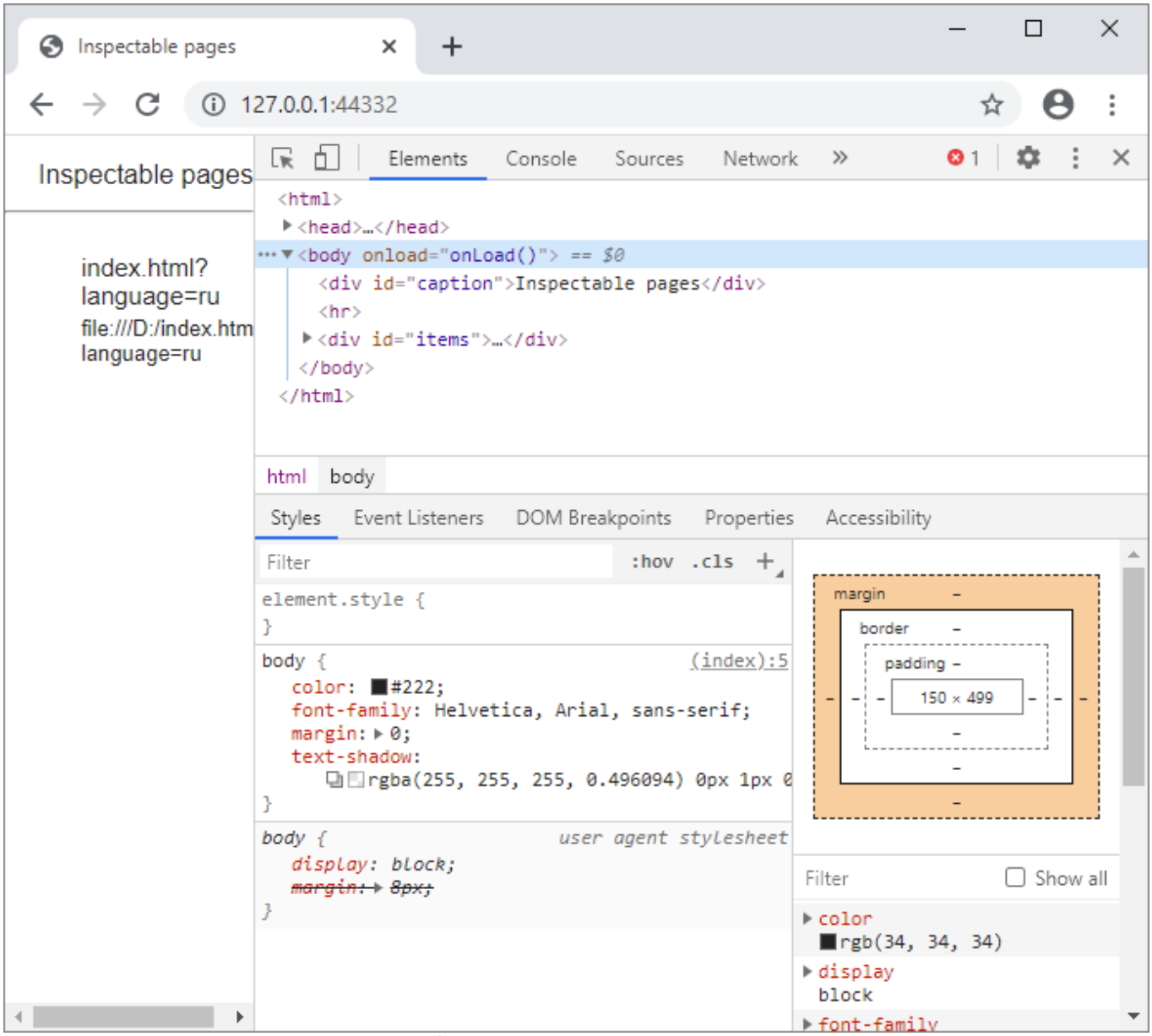

Рис. 5. Окно удаленной отладки приложения

3.3.2 Язык интерфейса внешнего приложения в HTML5-окне

По умолчанию, язык интерфейса внешнего приложения, запускаемого в окне *HTML5 FrontEnd* или *Всплывающем окне HTML5* совпадает с языком SecurOS. Если внешнее приложение поддерживает мультиязычность, изменить язык интерфейса можно с помощью параметра **URL** объекта (см. рис. 6).

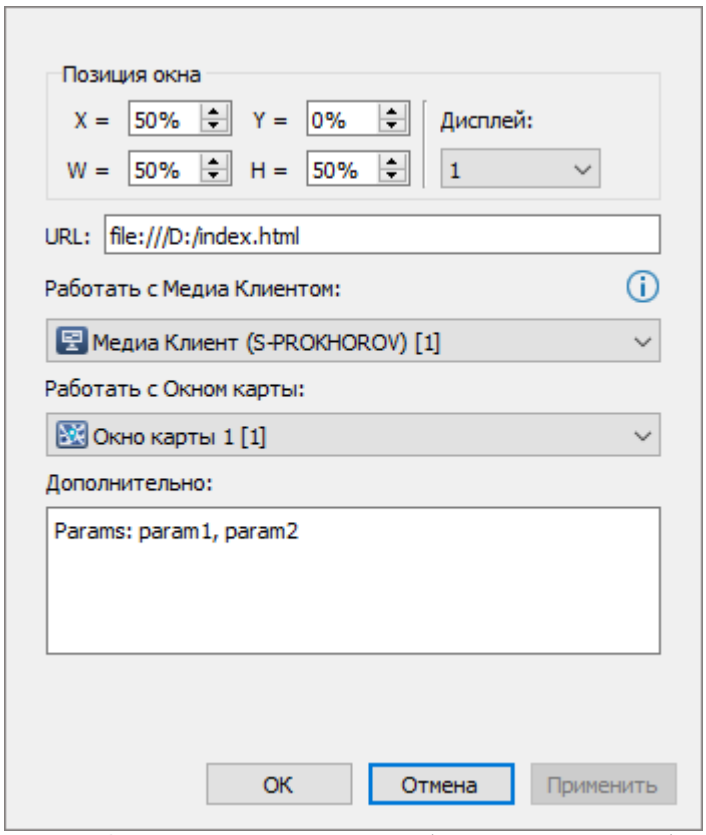

Рис. 6. Окно настройки параметров объекта HTML5 FrontEnd

Для изменения языка интерфейса внешнего приложения явным образом добавьте параметр `language` в значение параметра **URL**. Например:

```
file:///D:/index.html?language=ru
```

или

```
localhost:82?language=ru
```

Новое значение языка интерфейса внешнего приложения будет передано сразу после применения новых настроек.

Примечание. Для использования языка по умолчанию параметр `language` указывать необязательно.

---

4 Приложение 1. События и действия объектов SecurOS

В данном разделе описаны события и действия объектов подсистем SecurOS.

4.1 Основная подсистема

В данном разделе описаны события и действия объектов основной подсистемы.

4.1.1 Пользователь

Идентификатор типа объекта: `PERSON`.

Таблица 6. События `PERSON`

Идентификатор события: `CARD_DUPLICATED`	
Название в макрокоманде: Дублирование карты пользователя	
Описание: Дублирование карты пользователя	
Параметры: нет	
Идентификатор события: `REGISTERED`	
Название в макрокоманде: Регистрация пользователя	
Описание: Регистрация пользователя	
Параметры:	
`comment`	Структура, содержащая следующую информацию: • DNS-имя компьютера, с которого выполнено подключение, в формате `Клиент: <Имя>`; • DNS-имя компьютера, к которому выполнено подключение, в формате `Сервер: <Имя>`.
Идентификатор события: `UNREGISTERED`	
Название в макрокоманде: Завершение работы пользователя	
Описание: Завершение работы пользователя	
Параметры:	

comment	Структура, содержащая следующую информацию: • DNS-имя компьютера, с которого выполнено подключение, в формате Клиент: <Имя>; • DNS-имя компьютера, к которому выполнено подключение, в формате Сервер: <Имя>.
Идентификатор события: NEW_TICKET	
Название в макрокоманде: Карточка отправлена	
Описание: Карточка происшествия создана и отправлена	
Параметры:	
time	Время отправки *Карточки происшествия* оператором
owner	Идентификатор *Компьютера*, с которого отправлена *Карточка происшествия*
cam	Идентификатор *Камеры*, с помощью которой зафиксировано событие
comment	Структура, содержащая следующую информацию: • идентификатор *Карточки происшествия* в SecurOS • идентификатор и название *Протокола событий* или *Медиа Клиента*, с помощью которого была отправлена *Карточка происшествия*; • дата и время события (для *Протокола событий*) или кадра (для *Медиа Клиента*), на основе которого была создана *Карточка происшествия*; • адрес происшествия (в виде идентификатора и имени *Камеры*, с помощью которой было зафиксировано событие); • тип происшествия; • признак опасности для людей; • дополнительная информация (комментарий к событию).
date	Дата отправки *Карточки происшествия*
incident_time	Дата и время события (для *Протокола событий*) или кадра (для *Медиа Клиента*), на основе которого была создана *Карточка происшествия*
incident_type	Тип происшествия (по классификатору *Карточки происшествия*)
slave_id	Идентификатор *Компьютера*, с которого была отправлена *Карточка происшествия*
uuid	Идентификатор *Карточки происшествия* в SecurOS

Действия: отсутствуют.

4.1.2 Внешнее приложение

Идентификатор типа объекта: EXT_APP.

Таблица 7. События EXT_APP

Идентификатор события: APPLICATION_STARTED	
Название в макрокоманде: Приложение запущено	
Описание: Внешнее приложение успешно запущено	
Параметры:	
slave_id	Идентификатор компьютера, на котором запускалось внешнее приложение
date	Дата события, в формате ДД-ММ-ГГ
time	Время события, в формате ЧЧ:ММ:СС.XXX
Идентификатор события: APPLICATION_STOPPED	
Название в макрокоманде: Приложение остановило работу	
Описание: Внешнее приложение завершено	
Параметры:	
comment	Комментарий для *Протокола событий*, в формате Код возврата:<код_возврата>
slave_id	Идентификатор компьютера, на котором запускалось внешнее приложение
date	Дата события, в формате ДД-ММ-ГГ
time	Время события, в формате ЧЧ:ММ:СС.XXX
Идентификатор события: APPLICATION_FAILED	
Название в макрокоманде: Ошибка запуска приложения	
Описание: Запуск приложения завершился ошибкой	
Параметры:	
comment	Комментарий для *Протокола событий*, в формате Код возврата:<код_возврата>
slave_id	Идентификатор компьютера, на котором запускалось внешнее приложение
date	Дата события, в формате ДД-ММ-ГГ
time	Время события, в формате ЧЧ:ММ:СС.XXX

Действия: отсутствуют.

4.2 Подсистема интерфейса

В данном разделе описаны события и действия объектов подсистемы интерфейса.

4.2.1 Рабочий стол

Идентификатор типа объекта: `DISPLAY`.

События: отсутствуют.

Таблица 8. Действия `DISPLAY`

Идентификатор действия: `ACTIVATE`
Название в макрокоманде: Показать
Описание: Активировать *Рабочий стол* с заданным идентификатором
Внимание! Идентификатор *Рабочего стола* необходимо задавать с учетом идентификатора *Объекта охраны*, которому принадлежит данный *Рабочий стол*. Например, `DISPLAY|1.4|ACTIVATE`.
Параметры: нет
Идентификатор действия: `DEACTIVATE`
Название в макрокоманде: Скрыть
Описание: Скрыть *Рабочий стол* с заданным идентификатором
Внимание! Идентификатор *Рабочего стола* необходимо задавать с учетом идентификатора *Объекта охраны*, которому принадлежит данный *Рабочий стол*. Например, `DISPLAY|1.4|DEACTIVATE`.
Параметры: нет

4.2.2 Карта: Интерфейс оператора

Идентификатор типа объекта: `MAP3`.

События: отсутствуют.

Таблица 9. Действия `MAP3`

Идентификатор действия: `FIND_OBJECT`	
Название в макрокоманде: нет	
Описание: Переход к объекту на *Карте* — отображение уровня, на котором расположен объект и подсветка объекта	
Параметры:	
`type`	Тип искомого объекта

id	Уникальный идентификатор искомого объекта

4.2.3 Протокол событий

Идентификатор типа объекта: EVENT_VIEWER.

Таблица 10. События EVENT_VIEWER

Идентификатор события: ACTIVATE_OBJECT	
Название в макрокоманде: нет	
Описание: Генерируется при двойном нажатии на строку в протоколе событий	
Параметры:	
event_objid	Идентификатор объекта, отображаемого в строке протокола событий
event_objtype	Тип объекта, отображаемого в строке протокола событий
event_time	Время события, зафиксированное в протоколе событий (в формате ЧЧ:ММ:СС)
event_date	Дата события, зафиксированная в протоколе событий (в формате ДД-ММ-ГГ)

Действия: отсутствуют.

4.2.4 СКУД/ОПС: Интерфейс оператора

Идентификатор типа объекта: ACS_CLIENT.

События: отсутствуют.

Таблица 11. Действия `ACS_CLIENT`

Идентификатор действия: `APPLY_FILTER`	
Название в макрокоманде: нет	
Описание: Отображение временного фильтра по тем объектам СКУД/ОПС, которые указаны в команде	
Параметры:	
`filter_name`	Имя фильтра, выводимое в окне объекта *СКУД/ОПС: Интерфейс оператора*

`objects`	Список фильтруемых объектов СКУД/ОПС, в формате JSON. В структуре JSON-файла список задается в следующем формате:
	`{"<тип_объекта>": ["id_объекта_в_SecurOS", ...], ...}`
	Например:
	`{"BOLID": ["1","4"],"ENTRANCE": ["1","8"]}`

4.3 Видеоподсистема

В данном разделе описаны события и действия объектов видеоподсистемы.

4.3.1 Камера

Идентификатор типа объекта: `CAM`.

4.3.1.1 События

В разделе описаны следующие события *Камеры*:

- **ARMED**.
- **ATTACH**.
- **BLINDING**.
- **BROADCAST_REQUEST_PTZ**.
- **CAPABILITIES**.
- **DEFOCUSED**.
- **DETACH**.
- **DISARMED**.
- **FOCUSED**.
- **LIGHT_OFF_COMPLETED**.
- **LIGHT_ON_COMPLETED**.
- **MD_START**.
- **MD_STOP**.
- **PTZ_STATUS**.
- **REC**.

- **REC_ERROR**.
- **REC_STOP**.
- **SPEAKER_STATE_ACQUIRED**.
- **SPEAKER_STATE_READY**.
- **TELEMETRY_BUSY**.
- **TELEMETRY_RELEASED**.
- **UNBLINDING**.
- **VCA_EVENT**.
- **События операций переноса файлов архива**.

4.3.1.1.1 ARMED

Описание: Камера была поставлена на охрану командой (действием) `ARM` от объекта *Зона*.

Примечание. При установке камеры на охрану при помощи *Медиа Клиента* (кнопка **Поставить на охрану/Снять с охраны**) ставятся на охрану зоны, как следствие срабатывает событие `ARMED`.

Название в макрокоманде: Камера была поставлена на охрану

Параметры: нет

4.3.1.1.2 ATTACH

Описание: Поступил сигнал от камеры

Название в макрокоманде: Подключение

Параметры: нет

4.3.1.1.3 BLINDING

Описание: Камера засвечена

Название в макрокоманде: Плохая видимость

Параметры: нет

4.3.1.1.4 BROADCAST_REQUEST_PTZ

Идентификатор события: `BROADCAST_REQUEST_PTZ`

Название в макрокоманде: нет

Описание: Широковещательный запрос на освобождение захваченного управления PTZ. После отправки запроса на компьютерах всех *Пользователей* сети, которые могут работать с захваченной *Камерой*, отображается всплывающее окно с параметрами запроса.	
Параметры:	
`user_id`	Имя *Пользователя*, отправившего запрос на освобождение управления PTZ.
`user_host`	DNS-имя *Компьютера*, с которого поступил запрос на освобождение управления PTZ.
`hold_ptz_module_id`	IP-адрес или DNS-имя *Компьютера*, на котором захвачено управление PTZ, и название модуля, с помощью которого произошел захват, в формате `<имя_хоста>.<название_модуля>`. Например, `<E-PETROV.MediaClient>`.
`hold_ptz_owner_id`	Имя *Пользователя*, захватившего управление PTZ.

4.3.1.1.5 CAPABILITIES

Описание: Возможности камеры (посылается в ответ на команду **GET_CAPABILITIES**)	
Название в макрокоманде: нет	
Параметры: **Примечание.** Отсутствие любого из перечисленных параметров интерпретируется системой как получение значения false.	
`move_abs`	Поддержка позиционирования по абсолютным координатам. Возможные значения: • `true` — абсолютные координаты поддерживаются; • `false` — абсолютные координаты не поддерживаются.
`fisheye_lens`	Наличие объектива "рыбий глаз". Возможные значения: • `true` — объектив "рыбий глаз" установлен; • `false` — объектив "рыбий глаз" отсутствует.
`motorized_lens`	Наличие моторизированного объектива (управление PTZ имеет ограниченный функционал: приближение/удаление изображения). Возможные значения: • `true` — моторизированный объектив установлен; • `false` — моторизированный объектив отсутствует.

4.3.1.1.6 DEFOCUSED

Описание: Камера расфокусирована, фокус потерян

Название в макрокоманде: Потеря фокусировки

Параметры: нет

4.3.1.1.7 DETACH

Описание: Пропал сигнал от камеры

Название в макрокоманде: Обрыв

Параметры: нет

4.3.1.1.8 DISARMED

Описание: Камера была снята с охраны командой (действием) `DISARM` от объекта *Зона*.

Примечание. При снятии камеры с охраны при помощи *Медиа Клиента* (кнопка **Поставить на охрану/Снять с охраны**) снимаются с охраны зоны, как следствие срабатывает событие `DISARMED`.

Название в макрокоманде: Камера была снята с охраны

Параметры: нет

4.3.1.1.9 FOCUSED

Описание: Камера сфокусирована, изображение нормализовано

Название в макрокоманде: Сфокусирована

Параметры: нет

4.3.1.1.10 LIGHT_OFF_COMPLETED

Описание: Встроенное освещение камеры выключено

Название в макрокоманде: Освещение выключено

Параметры: нет

4.3.1.1.11 LIGHT_ON_COMPLETED

Описание: Встроенное освещение камеры включено

Название в макрокоманде: Освещение включено
Параметры: нет

4.3.1.1.12 MD_START

Описание: Генерируется при срабатывании датчика движения тревожной камеры
Название в макрокоманде: Тревога
Параметры: нет

4.3.1.1.13 MD_STOP

Описание: Движение прекратилось
Название в макрокоманде: Конец тревоги
Параметры: нет

4.3.1.1.14 PTZ_STATUS

Описание: Текущие координаты `PAN`, `TILT`, `ZOOM` камеры (полученные в ответ на команду `GET_PTZ_STATUS`)	
Название в макрокоманде: нет	
Параметры:	
`error`	Возможные значения: • `1` — запрос координат завершился с ошибкой (текущие PTZ-координаты камеры получить не удалось); • `0` — запрос координат завершился успешно (текущие PTZ-координаты камеры получены). **Примечание.** Если принимает значение `true`, параметры отсутствуют.
`pan_position`	Текущая `pan`-координата камеры.
`tilt_position`	Текущая `tilt`-координата камеры.
`zoom_position`	Текущая `zoom`-координата камеры.
`pan_position_norm`	Текущая нормализованная `pan`-координата камеры в координатном пространстве SecurOS.
`tilt_position_norm`	Текущая нормализованная `tilt`-координата камеры в координатном пространстве SecurOS.

zoom_position_norm	Текущая нормализованная zoom-координата камеры в координатном пространстве SecurOS.

Примечание. Нормализованные координаты могут быть использованы в качестве параметров для действия **MOVE_ABS**.

4.3.1.1.15 REC

Описание: Начало записи на жесткий диск

Название в макрокоманде: Запись на диск

Параметры: нет

4.3.1.1.16 REC_ERROR

Описание: Ошибка записи на диск

Название в макрокоманде: Ошибка записи

Параметры: нет

4.3.1.1.17 REC_STOP

Описание: Конец записи на диск

Название в макрокоманде: Конец записи на диск

Параметры: нет

4.3.1.1.18 SPEAKER_STATE_ACQUIRED

Описание: Динамик камеры включен

Название в макрокоманде: Трансляция звука на динамик включена

Параметры:

comment	Ссылка на *Компьютер*, оператор которого включил динамик *Камеры* (нажата кнопка **Включить динамик**), в формате Компьютер<пробел><ID_объекта_в_SecurOS>

4.3.1.1.19 SPEAKER_STATE_READY

Описание: Динамик камеры отключен

Название в макрокоманде: Трансляция звука на динамик выключена	
Параметры:	
comment	Ссылка на *Компьютер*, оператор которого выключил динамик *Камеры* (нажата кнопка **Выключить динамик**), в формате Компьютер<пробел><ID_объекта_в_SecurOS>

4.3.1.1.20 TELEMETRY_BUSY

Описание: Управление телеметрией данной *Камеры* заблокировано	
Название в макрокоманде: нет	
Параметры:	
SLAVE_ID	Идентификатор *Компьютера*, с которого была отправлена команда управления PTZ, блокировавшая управление.
owner_id	Идентификатор *Пользователя*, заблокировавшего управление PTZ (поле **Название** объекта *Пользователь*).
priority	Приоритет *Пользователя*, заблокировавшего управление PTZ.
permanently	Метод захвата управления (с длительным удержанием/с автоматическим освобождением). Возможные значения: • true — управление захвачено по команде TELEMETRY_ACQUIRE в режиме длительного удержания управления. • false — управление захвачено по любой команде, отличной от TELEMETRY_ACQUIRE, в режиме раздельного управления без длительного удержания.
time	Время блокировки управления, в формате ЧЧ:ММ:СС.XXX.
date	Дата блокировки управления, в формате ДД-ММ-ГГ.
hold_ptz_allow_display_user	Признак отображения имени пользователя, захватившего телеметрию (заданного в параметре owner_id). Возможные значения: • true — имя отображается; • false — имя не отображается.

4.3.1.1.21 TELEMETRY_RELEASED

Описание: Управление телеметрией данной *Камеры* разблокировано
Название в макрокоманде: нет
Параметры:

slave_id	Идентификатор *Компьютера*, с которого была отправлена команда разблокировки управления PTZ.
permanently	Метод освобождения управления (из режима с длительным удержанием/из режима с автоматическим освобождением). Возможные значения: • true — управление освобождено по команде **TELEMETRY_RELEASE** в режиме длительного удержания управления. • false — управление освобождено по любой команде, отличной от **TELEMETRY_RELEASE**, в режиме раздельного управления без длительного удержания.
time	Время разблокировки управления, в формате ЧЧ:ММ:СС.XXX.
date	Дата разблокировки управления, в формате ДД-ММ-ГГ.

4.3.1.1.22 UNBLINDING

Описание: Камера открыта
Название в макрокоманде: Хорошая видимость
Параметры: нет

4.3.1.1.23 VCA_EVENT

Описание: Событие видеоаналитики	
Название в макрокоманде: Событие видеоаналитики	
Параметры:	

comment	Структура JSON, использующаяся для вывода информации о событии в *Протоколе событий*. Может иметь следующий формат:
	{
	"comment": <commentText>,
	"visualization": <visualization>,
	"description": <description>
	}

commentText	Текстовый комментарий, отображаемый в *Протоколе событий*. Может включать в себя следующие строки: • Пустую строку, либо одну и более строк с комментарием. Формат и наличие зависит от типа и настроек детектора. • Приоритет зоны. Наличие зависит от типа детектора. • Приоритет события. Задается в настройках детектора.
date	Дата генерации события.
description	Название события, заданное в настройках детектора. Отображается в *Протоколе событий*.

`event_priority`	Приоритет события, заданный в настройках детектора. Отображается в *Протоколе событий*. Возможные значения: [`1; 10`].
`plugin`	Название плагина, детектор которого сгенерировал событие: • Если событие сгенерировано детектором плагина *Tracking Kit III*, всегда принимает значение `IV^Engine`. • Если событие сгенерировано независимым детектором, принимает одно из следующих значений: – `Helmet^Engine` — детектор людей без касок. – `Smoke^Engine` — детектор дыма. – `Fight^Engine` — нейросетевой детектор драк. – `FireSmoke^Engine` — нейросетевой детектор дыма и огня. – `Loitering^Engine` — нейросетевой детектор праздношатания. – `ManDown^Engine` — нейросетевой детектор упавших людей. – `EstimationCrowd^Engine` — нейросетевой счетчик людей.
`source_id`	Идентификатор объекта *Компьютерное зрение* в *Дереве объектов* SecurOS.
`source_type`	Всегда принимает значение `TRACKINGKIT3`.
`time`	Время генерации события.
`type`	Идентификатор типа события. • Если событие сгенерировано детектором плагина *Tracking Kit III*, принимает одно из следующих значений: – `crowd` — детектор скопления людей. – `dwelling` — детектор пребывания в зоне. – `intrusion` — детектор проникновения в зону. – `leftObject` — детектор оставленных предметов. – `removedObject` — детектор унесенных предметов. – `crossCounter` — детектор пересечения линии. – `lineCross` — счетчик объектов. – `loitering` — детектор праздношатания. – `running` — детектор бега. – `wrongDirection` — детектор движения в запрещенном направлении. • Если событие сгенерировано независимым детектором, значение идентично значению параметра `plugin` (см. выше).

visualization	Набор параметров в формате команд модуля **ImageProcessor**. Используются для визуализации графики (рамок объектов, зон детекции, линий пересечения, направления движения и т. д.) при переходе из *Протокола событий* к просмотру кадра с событием в *Медиа Клиенте*.

Таблица 12. Возможные значения основных параметров события VCA_EVENT для камер Hanhwa

Параметр	Описание
plugin	Название источника события. Всегда принимает значение Hanhwa.
type	Идентификатор типа события. Возможные значения: • VA_ENTERING_ALARM — событие проникновения в зону. • VA_EXITING_ALARM — событие выхода из зоны. • VA_PASSING_ALARM — событие пересечения зоны. • VA_APPEARING_ALARM — событие появления в зоне. • VA_DISAPPEARING_ALARM — событие исчезновения из зоны. • MD_ALARM — событие движения в зоне.
description	Название события в *Протоколе событий*. Указывает на начало или окончание соответствующего события. Например, если проникновение в охраняемую зону зафиксировано в его начале, в параметре будет выведено следующее сообщение: Entering (Event Start)

4.3.1.1.24 События операций переноса файлов архива

В разделе описаны следующие события, возникающие при работе с файлами переносимого архива:

- **ARCH_FILES_OP_SCHEDULED**.
- **ARCH_FILES_OP_STARTED**.
- **ARCH_FILES_OP_UPDATED**.
- **ARCH_FILES_OP_DONE**.
- **ARCH_FILES_OP_ERROR**.
- **ARCH_FILES_OP_CANCELLED**.
- **ARCH_FILES_OP_CLEARED**.

4.3.1.1.24.1 ARCH_FILES_OP_SCHEDULED

Описание: Задача переноса архива добавлена в очередь задач
Название в макрокоманде: нет

Параметры:	
op_id	Идентификатор задачи, присвоенный при постановке в очередь
op_type	Тип задачи. Возможные значения: • move — перемещение архива; • erase — удаление перемещенного архива.
op_info	Структура в формате JSON, содержащая информацию о процессе выполнения задачи. Может содержать следующие параметры: • time_from — дата и время начала временного интервала для переноса архива; • time_until — дата и время окончания временного интервала для переноса архива; • op_user — идентификатор клиента/оператора, отправившего команду; • op_move_src — сервер, с которого перемещается архив; • op_move_dst — сервер, на который перемещается архив.
op_progress	Прогресс выполнения задачи (в процентах)
time	Время актуальности события (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX)

4.3.1.1.24.2 ARCH_FILES_OP_STARTED

Описание: Задача переноса архива начала выполняться	
Название в макрокоманде: нет	
Параметры:	
op_id	Идентификатор задачи, присвоенный при постановке в очередь
op_type	Тип задачи. Возможные значения: • move — перемещение архива; • erase — удаление перемещенного архива.

op_info	Структура в формате JSON, содержащая информацию о процессе выполнения задачи. Может содержать следующие параметры: • time_from — дата и время начала временного интервала для переноса архива; • time_until — дата и время окончания временного интервала для переноса архива; • op_user — идентификатор клиента/оператора, отправившего команду; • op_move_src — сервер, с которого перемещается архив; • op_move_dst — сервер, на который перемещается архив.
op_progress	Прогресс выполнения задачи (в процентах)
time	Время актуальности события (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX)

4.3.1.1.24.3 ARCH_FILES_OP_UPDATED

Описание: Периодическое событие о прогрессе выполнения задачи переноса архива

Примечание. По умолчанию событие отправляется каждые 5 минут.

Название в макрокоманде: нет

Параметры:

op_id	Идентификатор задачи, присвоенный при постановке в очередь
op_type	Тип задачи. Возможные значения: • move — перемещение архива; • erase — удаление перемещенного архива.
op_info	Структура в формате JSON, содержащая информацию о процессе выполнения задачи. Может содержать следующие параметры: • time_from — дата и время начала временного интервала для переноса архива; • time_until — дата и время окончания временного интервала для переноса архива; • op_user — идентификатор клиента/оператора, отправившего команду; • op_move_src — сервер, с которого перемещается архив; • op_move_dst — сервер, на который перемещается архив.
op_progress	Прогресс выполнения задачи (в процентах)

time	Время актуальности события (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX)

4.3.1.1.24.4 ARCH_FILES_OP_DONE

Описание: Задача переноса архива выполнена успешно	
Название в макрокоманде: нет	
Параметры:	
op_id	Идентификатор задачи, присвоенный при постановке в очередь
op_type	Тип задачи. Возможные значения: • move — перемещение архива; • erase — удаление перемещенного архива.
op_info	Структура в формате JSON, содержащая информацию о процессе выполнения задачи. Может содержать следующие параметры: • time_from — дата и время начала временного интервала для переноса архива; • time_until — дата и время окончания временного интервала для переноса архива; • op_user — идентификатор клиента/оператора, отправившего команду; • op_move_src — сервер, с которого перемещается архив; • op_move_dst — сервер, на который перемещается архив.
op_progress	Прогресс выполнения задачи (в процентах)
time	Время актуальности события (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX)

4.3.1.1.24.5 ARCH_FILES_OP_ERROR

Описание: При выполнении задачи переноса архива произошла ошибка	
Название в макрокоманде: нет	
Параметры:	
op_id	Идентификатор задачи, присвоенный при постановке в очередь

op_type	Тип задачи. Возможные значения: • move — перемещение архива; • erase — удаление перемещенного архива.
op_info	Структура в формате JSON, содержащая информацию о процессе выполнения задачи. Может содержать следующие параметры: • time_from — дата и время начала временного интервала для переноса архива; • time_until — дата и время окончания временного интервала для переноса архива; • op_user — идентификатор клиента/оператора, отправившего команду; • op_move_src — сервер, с которого перемещается архив; • op_move_dst — сервер, на который перемещается архив.
op_progress	Прогресс выполнения задачи (в процентах)
error	Числовой код ошибки
message	Описание ошибки

4.3.1.1.24.6 ARCH_FILES_OP_CANCELLED

Описание: Задача переноса архива отменена пользователем или командой/скриптом

Примечание. Событие отправляется при штатной отмене задачи, не связанной с потерей связи с *Удаленной системой*, сменой пароля *Удаленной системы* или ошибками в ходе выполнения задачи.

Название в макрокоманде: нет

Параметры:

op_id	Идентификатор задачи, присвоенный при постановке в очередь
op_type	Тип задачи. Возможные значения: • move — перемещение архива; • erase — удаление перемещенного архива.

op_info	Структура в формате JSON, содержащая информацию о процессе выполнения задачи. Может содержать следующие параметры: • time_from — дата и время начала временного интервала для переноса архива; • time_until — дата и время окончания временного интервала для переноса архива; • op_user — идентификатор клиента/оператора, отправившего команду; • op_move_src — сервер, с которого перемещается архив; • op_move_dst — сервер, на который перемещается архив.
op_progress	Прогресс выполнения задачи (в процентах)
time	Время актуальности события (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX)

4.3.1.1.24.7 ARCH_FILES_OP_CLEARED

Описание: Задача переноса архива удалена из списка выполненных задач	
Название в макрокоманде: нет	
Параметры:	
op_id	Идентификатор задачи, присвоенный при постановке в очередь
op_type	Тип задачи. Возможные значения: • move — перемещение архива; • erase — удаление перемещенного архива.
op_info	Структура в формате JSON, содержащая информацию о процессе выполнения задачи. Может содержать следующие параметры: • time_from — дата и время начала временного интервала для переноса архива; • time_until — дата и время окончания временного интервала для переноса архива; • op_user — идентификатор клиента/оператора, отправившего команду; • op_move_src — сервер, с которого перемещается архив; • op_move_dst — сервер, на который перемещается архив.
op_progress	Прогресс выполнения задачи (в процентах)

`time`	Время актуальности события (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`)

4.3.1.2 Действия

Для удобства пользователя команды управления *Камерой* разделены на следующие группы:

- **Информация об аппартных возможностях камеры**.
- **Работа с видео**.
- **Управление фокусным расстоянием камеры**.
- **Управление диафрагмой камеры**.
- **Управление движением камеры**.
- **Работа с Препозициями и Турами**.
- **Состояние поворотного устройства**.
- **Управление дополнительными устройствами камеры**.
- **Команды для постановки задач переноса архива**.
- **Команды управления задачами переноса архива**.

Внимание! Набор поддерживаемых команд зависит от **Типа** и **Модели** *Камеры*.

4.3.1.2.1 Информация об аппартных возможностях камеры

В разделе описаны следующие команды работы с информацией об аппартаных возможностях камеры:

- **GET_CAPABILITIES**.

4.3.1.2.1.1 GET_CAPABILITIES

Описание: Запрос аппаратных возможностей камеры. Список возможностей вернется в событии **CAPABILITIES**
Название в макрокоманде: нет
Параметры: нет

4.3.1.2.2 Работа с видео

В разделе описаны следующие команды работы с видео:

- **ADD_SUBTITLES**.
- **ARM**.
- **CLEAR_SUBTITLES**.
- **DISARM**.
- **REC**.
- **REC_ROLLBACK**.
- **REC_STOP**.

- **REQUEST_MASK**.
- **SET_MUX**.
- **START_VIDEO**.
- **STOP_VIDEO**.

4.3.1.2.2.1 ADD_SUBTITLES

Описание: Добавить субтитры в видеопоток	
Название в макрокоманде: нет	
Параметры:	
`command`	Текст субтитров. **Примечание.** Для форматирования текста субтитров внутри параметра могут использоваться HTML-тэги работы с текстом (национальные алфавиты, размер и цвет шрифта, начертание, перенос по строкам и т.д.). В тексте субтитров также могут быть использованы специальные Unicode-символы. Подробнее см. подраздел **Работа с субтитрами**. **Внимание!** Для отображения субтитров в режимах **Он-лайн/Архив** отметьте флажок **Отображать субтитры** на вкладке **Камеры** в настройках объекта *Медиа Клиент* (см. **Руководство администратора SecurOS**). В противном случае субтитры выводится не будут (на запись субтитров в видеопоток настройка не влияет). При выводе субтитров приоритет имеют параметры форматирования, заданные в команде.

4.3.1.2.2.2 ARM

Описание: Поставить камеру на охрану	
Название в макрокоманде: Поставить на охрану	
Параметры: нет	

4.3.1.2.2.3 CLEAR_SUBTITLES

Описание: Удалить субтитры из видеопотока **Примечание.** Подробнее об удалении субтитров см. подраздел **Работа с субтитрами**.
Название в макрокоманде: нет
Параметры: нет

4.3.1.2.2.4 DISARM

Описание: Снять камеру с охраны

Название в макрокоманде: Снять с охраны
Параметры: нет

4.3.1.2.2.5 REC

Описание: Начать запись **Внимание!** Команда будет проигнорирована, если *Камера* работает в режиме `Запретить запись` (см. **Руководство администратора SecurOS**).	
Название в макрокоманде: Начать запись	
Параметры:	
`hot_rec_time`	Время, в течение которого запись ведется с исходным FPS потока, в миллисекундах
`priority`	Приоритет команды. Возможные значения: • `0` — запись после перезапуска системы не продолжается, в процессе работы запись может быть остановлена командой как с приоритетом `1`, так и `0`. Данное значение приоритета устанавливается по умолчанию при отправке команды с помощью *Макрокоманды* или *Скрипта Node.js*. При отправке указанными способами значение приоритета может быть переопределено на `1`. • `1` — запись после перезапуска системы продолжается, в процессе работы запись может быть остановлена только командой с приоритетом `1`. Данное значение приоритета устанавливается при отправке команды оператором из окна *Медиа Клиента*, окна *Карта: Интерфейс оператора* или окна *ГИС: Интерфейс оператора*. Оператор также может отправить команду: • из окна модуля WebConnect (см. **Краткое руководство пользователя SecurOS WebConnect**); • с помощью запроса REST API (см. **Руководство программиста SecurOS REST API**); • с помощью CCTV-клавиатуры (см. **Руководство администратора SecurOS**). Во всех указанных случаях команда будет отправлена с приоритетом `0`.

4.3.1.2.2.6 REC_ROLLBACK

Описание: Начать запись, в начало которой будет добавлено видео из буфера предзаписи

Примечание. Длительность предзаписи задается в настройках объекта *Камера*, см. **Руководство администратора SecurOS**.

Внимание! Команда будет проигнорирована, если *Камера* работает в режиме `Запретить запись` (см. **Руководство администратора SecurOS**).

Название в макрокоманде: Начать запись с откатом

Параметры:

`rollback_time_abs`	Абсолютное время на временной шкале, в формате `ЧЧ:ММ:СС.XXX`, для текущей даты. Влияет на размер добавляемого фрагмента буфера предзаписи или указывает на время сохраняемого кадра в зависимости от режима работы команды (см. описание параметра `start_rec`). Полный размер буфера предзаписи устанавливается в настройках объекта *Камера*, см. **Руководство администратора SecurOS**.
`hot_rec_time`	Время, в течение которого запись ведется с исходным FPS потока, в миллисекундах

`start_rec`	Режим записи — кадр или видеофрагмент. • `1` или `не задан` — сохранение видеофрагмента. – если параметр `rollback_time_abs` не задан или заданное значение находится за правой границей временного интервала предзаписи — буфер предзаписи не добавляется; – если заданное значение параметра `rollback_time_abs` находится внутри временного интервала предзаписи — будет добавлен фрагмент буфера предзаписи, начиная с указанного времени; – если заданное значение параметра `rollback_time_abs` находится за левой границей временного интервала предзаписи — буфер предзаписи будет добавлен полностью. • `0` — сохранение кадра. – если параметр `rollback_time_abs` не задан или заданное значение находится за правой границей временного диапазона буфера предзаписи, сохранится текущий кадр видео; – если заданное значение параметра `rollback_time_abs` лежит в пределах временного интервала буфера предзаписи, сохранится кадр, время которого соответствует заданному значению параметра `rollback_time_abs`; – если заданное значение параметра `rollback_time_abs` находится за левой границей временного интервала буфера предзаписи, кадр сохраняться не будет.
`priority`	Приоритет команды. Возможные значения: • `0` — запись после перезапуска системы не продолжается, в процессе работы запись может быть остановлена командой как с приоритетом `1`, так и `0`. Данное значение приоритета устанавливается по умолчанию при отправке команды с помощью *Макрокоманды* или *Скрипта Node.js*. При отправке указанными способами значение приоритета может быть переопределено на `1`. • `1` — запись после перезапуска системы продолжается, в процессе работы запись может быть остановлена только командой с приоритетом `1`. Данное значение приоритета устанавливается при отправке команды оператором из окна *Медиа Клиента*, окна *Карта: Интерфейс оператора* или окна *ГИС: Интерфейс оператора*.

4.3.1.2.2.7 REC_STOP

Описание: Остановка записи

Внимание! Команда будет проигнорирована, если *Камера* работает в режиме `Непрерывной записи` (см. **Руководство администратора SecurOS**).

Название в макрокоманде: Остановка записи

Параметры:

`priority`	Приоритет команды. Возможные значения: • `0 или не задан` — запись будет остановлена, если камера была поставлена на запись с приоритетом `0`. Данное значение приоритета устанавливается по умолчанию при отправке команды с помощью *Макрокоманды* или *Скрипта Node.js*. При отправке указанными способами значение приоритета может быть переопределено на `1`. • `1` — запись будет остановлена вне зависимости от того, с каким приоритетом камера была поставлена на запись. Данное значение приоритета устанавливается при отправке команды оператором из окна *Медиа Клиента*, окна *Карта: Интерфейс оператора* или окна *ГИС: Интерфейс оператора*. Оператор также может отправить команду: • из окна модуля WebConnect (см. **Краткое руководство пользователя SecurOS WebConnect**); • с помощью запроса REST API (см. **Руководство программиста SecurOS REST API**); • с помощью CCTV-клавиатуры (см. **Руководство администратора SecurOS**). Во всех указанных случаях команда будет отправлена с приоритетом `0`.

4.3.1.2.2.8 REQUEST_MASK

Описание: Запрос маски

Название в макрокоманде: нет

Параметры:

`mask`	Маска

4.3.1.2.2.9 SET_MUX

Описание: Переключить воспроизведение видеопотока на указанный канал. **Внимание!** Применяется только для *Устройств видеозахвата* с типом **Player AVI**.	
Название в макрокоманде: нет	
Параметры:	
`mux`	Номер канала

4.3.1.2.2.10 START_VIDEO

Описание: Запустить видеопоток с текущей камеры	
Название в макрокоманде: нет	
Параметры:	
`slave_id`	Имя компьютера, к которому подключена камера
`compress`	Необходимость использовать сжатие (`0` — не использовать, тогда значение параметра `compression` будет проигнорировано, `1` — использовать)
`compression`	Степень сжатия (`0` — отсутствует, `1` — максимальное качество, . . . , `5` — минимальное качество)
`speed`	Желаемая скорость видеопотока в кадрах в секунду (для принудительного ограничения)
`codec`	Полное имя (с расширением) XML-файла с профилем кодека. Если указан несуществующий файл или ничего не указано, то будет спользоваться текущий профиль камеры на сервере (например, `"codec","mjpeg.xml"`)

4.3.1.2.2.11 STOP_VIDEO

Описание: Выключить видеопоток с текущей камеры	
Название в макрокоманде: нет	
Параметры:	
`slave_id`	Имя компьютера, к которому подключена камера

4.3.1.2.3 Управление фокусным расстоянием камеры

В разделе описаны следующие команды управления фокусным расстоянием камеры:

- **FOCUS_AUTO_MODE**.
- **FOCUS_AUTO_MODE_DISABLE**.

- **FOCUS_IN**.
- **FOCUS_OUT**.
- **FOCUS_STOP**.

4.3.1.2.3.1 FOCUS_AUTO_MODE

Описание: Автофокусировка	
Название в макрокоманде: нет	
Параметры:	
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.3.2 FOCUS_AUTO_MODE_DISABLE

Описание: Отключить автофокусировку	
Название в макрокоманде: нет	
Параметры:	
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.3.3 FOCUS_IN

Описание: Увеличить фокусное расстояние объектива **Примечание.** Изменение фокусного расстояния (движение объектива) осуществляется непрерывно до получения команды **FOCUS_STOP**.	
Название в макрокоманде: нет	
Параметры:	
`speed`	Скорость изменения фокусного расстояния. Возможные значения: [`1; 10`]. Опциональный параметр
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.3.4 FOCUS_OUT

Описание: Уменьшить фокусное расстояние объектива **Примечание.** Изменение фокусного расстояния (движение объектива) осуществляется непрерывно до получения команды **FOCUS_STOP**.	
Название в макрокоманде: нет	
Параметры:	
`speed`	Скорость изменения фокусного расстояния. Возможные значения: [`1; 10`]. Опциональный параметр
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.3.5 FOCUS_STOP

Описание: Остановить изменение фокусного расстояния объектива	
Название в макрокоманде: нет	
Параметры:	
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.4 Управление диафрагмой камеры

В разделе описаны следующие команды управления диафрагмой камеры:

- **IRIS_AUTO_MODE**.
- **IRIS_CLOSE**.
- **IRIS_OPEN**.
- **IRIS_STOP**.

4.3.1.2.4.1 IRIS_AUTO_MODE

Описание: Автодиафрагма	
Название в макрокоманде: нет	
Параметры:	
`user_id`	Имя пользователя. Необязательный параметр

priority	Приоритет управления PTZ для данного пользователя. Возможные значения: [1; 10]. Значение по умолчанию — 1.

4.3.1.2.4.2 IRIS_CLOSE

Описание: Закрыть диафрагму

Примечание. Закрытие диафрагмы осуществляется непрерывно до получения команды **IRIS_STOP**.

Название в макрокоманде: нет

Параметры:

speed	Скорость закрытия диафрагмы. Возможные значения: [1; 10]. Значение по умолчанию — 1.
user_id	Имя пользователя. Необязательный параметр
priority	Приоритет управления PTZ для данного пользователя. Возможные значения: [1; 10]. Значение по умолчанию — 1.

4.3.1.2.4.3 IRIS_OPEN

Описание: Открыть диафрагму

Примечание. Открытие диафрагмы осуществляется непрерывно до получения команды **IRIS_STOP**.

Название в макрокоманде: нет

Параметры:

speed	Скорость открытия диафрагмы. Возможные значения: [1; 10]. Значение по умолчанию — 1.
user_id	Имя пользователя. Необязательный параметр
priority	Приоритет управления PTZ для данного пользователя. Возможные значения: [1; 10]. Значение по умолчанию — 1.

4.3.1.2.4.4 IRIS_STOP

Описание: Остановить изменение диафрагмы

Название в макрокоманде: нет

Параметры:

user_id	Имя пользователя. Необязательный параметр

`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.5 Управление движением камеры

В разделе описаны следующие команды управления движением камеры:

- **AREAZOOM**.
- **CENTER**.
- **HORIZONTAL_STOP**.
- **MOVE**.
- **MOVE_ABS**.
- **MOVE_STOP**.
- **VERTICAL_STOP**.

4.3.1.2.5.1 AREAZOOM

Описание: Масштабировать выделенную область кадра на всю площадь кадра	
Название в макрокоманде: нет	
Параметры:	
`x1, y1`	Координаты левого верхнего угла выделяемой области, в долях от размера кадра. Диапазон возможных значений: [`0; 10`]. **Примечание.** Начало координат (`0,0`) располагается в левом верхнем углу кадра.
`x2, y2`	Координаты правого нижнего угла выделяемой области, в долях от размера кадра. Диапазон возможных значений: [`0; 10`]. **Примечание.** Начало координат (`0,0`) располагается в левом верхнем углу кадра.

4.3.1.2.5.2 CENTER

Описание: Установить центр камеры в точку с заданными координатами	
Название в макрокоманде: нет	
Параметры:	
`x, y`	Координаты нового центра камеры, в долях от размера кадра. Диапазон возможных значений: [`0; 10`]. **Примечание.** Начало координат (`0,0`) располагается в левом верхнем углу кадра.

4.3.1.2.5.3 HORIZONTAL_STOP

Описание: Остановить поворот камеры по горизонтали
Название в макрокоманде: нет
Параметры: нет

4.3.1.2.5.4 MOVE

Описание: Переместить камеру в заданном направлении и/или изменить масштаб

Примечание: Перемещение в указанном направлении или масштабирование осуществляется непрерывно до тех пор, пока:

1. не получена команда остановки **MOVE_STOP**.
2. не получена команда `MOVE` с нулевыми значениями параметров (`pan_speed<0>`, `tilt_speed<0>`, `zoom_speed<0>`).
3. не достигнуты пороговые значения угла наклона камеры или увеличения. В этом случае остановка выполняется автоматически.

Название в макрокоманде: нет

Параметры:

`pan_speed`	Переместить камеру по горизонтали с указанной скоростью (`"-X"` — налево, `"X"` — направо)
`tilt_speed`	Переместить камеру по вертикали с указанной скоростью (`"-Y"` — вверх, `"Y"` — вниз)
`zoom_speed`	Изменить масштаб с указанной скоростью (`"-Z"` — увеличить, `"Z"` — уменьшить)
`duration`	Таймаут (в мс), по истечение которого перемещение/масштабирование камеры будет автоматически остановлено. Если не задан, перемещение/масштабирование камеры будет выполняться постоянно.
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.5.5 MOVE_ABS

Описание: Переместить камеру в заданное положение и/или изменить масштаб
Название в макрокоманде: нет
Параметры:

`pan`	Поворот камеры. Возможные значения: [`-10800; 10800`]

`tilt`	Наклон камеры. Возможные значения: [`-5400; 5400`]
`zoom`	Масштаб. Возможные значения: [`0; 1000`]

4.3.1.2.5.6 MOVE_STOP

Описание: Остановить изменение масштаба или перемещение камеры по вертикали или горизонтали
Название в макрокоманде: нет
Параметры: нет

4.3.1.2.5.7 VERTICAL_STOP

Описание: Остановить наклон камеры по вертикали
Название в макрокоманде: нет
Параметры: нет

4.3.1.2.6 Работа с Препозициями и Турами

В разделе описаны следующие команды работы с Препозициями и Турами:

- **CREATE_PRESET**.
- **HOME**.
- **PATROL_PLAY**.
- **PATROL_REMOVE**.
- **PATROL_STOP**.
- **PRESET_RECALL**.
- **REMOVE_PRESET**.
- **RENAME_PRESET**.
- **UPDATE_PRESET**.
- **HOME_SET**.

4.3.1.2.6.1 CREATE_PRESET

Описание: Создать препозицию	
Название в макрокоманде: нет	
Параметры:	
`name`	Название препозиции (допускается использование символов кириллического алфавита).

4.3.1.2.6.2 HOME

Описание: Перевести камеру в "домашнее" положение	
Название в макрокоманде: нет	
Параметры:	
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.6.3 PATROL_PLAY

Описание: Активировать маршрут патрулирования	
Название в макрокоманде: нет	
Параметры:	
`patrol`	Идентификатор маршрута, присвоенный SecurOS (значение `[ID]` из полного названия маршрута, отображаемого в формате `<Name><[ID]>` в списке маршрутов камеры в окне *Медиа Клиента*). **Примечание.** Если не задан, активируется первый маршрут патрулирования из списка маршрутов камеры.
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.6.4 PATROL_REMOVE

Описание: Удалить записанный тур	
Название в макрокоманде: нет	
Параметры:	
`patrol`	Идентификатор маршрута, присвоенный SecurOS (значение `[ID]` из полного названия маршрута, отображаемого в формате `<Name><[ID]>` в списке маршрутов камеры в окне *Медиа Клиента*).

4.3.1.2.6.5 PATROL_STOP

Описание: Остановить патрулирование
Название в макрокоманде: нет

Параметры:	
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.6.6 PRESET_RECALL

Описание: Перейти в указанную препозицию	
Название в макрокоманде: нет	
Параметры:	
`preset`	Идентификатор препозиции, присвоенный SecurOS (значение `[ID]` из полного названия препозиции, отображаемого в формате `<Name><[ID]>` в списке препозиций камеры в окне *Медиа Клиента*).
`pt_speed`	Скорость перемещения камеры в указанную препозицию. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.
`user_id`	Имя пользователя. Необязательный параметр
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.6.7 REMOVE_PRESET

Описание: Удалить препозицию	
Название в макрокоманде: нет	
Параметры:	
`id`	Идентификатор препозиции, присвоенный SecurOS (значение `[ID]` из полного названия препозиции, отображаемого в формате `<Name><[ID]>` в списке препозиций камеры в окне *Медиа Клиента*).

4.3.1.2.6.8 RENAME_PRESET

Описание: Переименовать препозицию	
Название в макрокоманде: нет	
Параметры:	
`id`	Идентификатор препозиции, присвоенный SecurOS (значение `[ID]` из полного названия препозиции, отображаемого в формате `<Name><[ID]>` в списке препозиций камеры в окне *Медиа Клиента*).

`name`	Новое название препозиции (допускается использование символов кириллического алфавита).

4.3.1.2.6.9 UPDATE_PRESET

Описание: Обновить препозицию (присвоить текущие координаты)	
Название в макрокоманде: нет	
Параметры:	
`id`	Идентификатор препозиции, присвоенный SecurOS (значение `[ID]` из полного названия препозиции, отображаемого в формате `<Name><[ID]>` в списке препозиций камеры в окне *Медиа Клиента*).

4.3.1.2.6.10 HOME_SET

Описание: Установить домашнюю препозицию
Название в макрокоманде: нет
Параметры: нет

4.3.1.2.7 Состояние поворотного устройства

В разделе описаны следующие команды управления состоянием поворотного устройства:

- **GET_PTZ_STATUS**.
- **REQUEST_PTZ**.
- **TELEMETRY_ACQUIRE**.
- **TELEMETRY_RELEASE**.

4.3.1.2.7.1 GET_PTZ_STATUS

Описание: Получить состояние PTZ	
Название в макрокоманде: нет	
Параметры:	
`__from`	Имя внешнего клиента, который отправил запрос на получение текущих координат камеры. Обязательный параметр. Координаты возвращаются в событии **PTZ_STATUS**.

4.3.1.2.7.2 REQUEST_PTZ

Описание: Отправить запрос на отключение удержания телеметрии на указанный *Компьютер*

Название в макрокоманде: нет	
Параметры:	
`user_id`	Идентификатор пользователя, которому будет отправлен запрос на отключение удержания телеметрии
`user_host`	Имя *Компьютера*, на который будет отправлен запрос
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.7.3 TELEMETRY_ACQUIRE

Описание: Включить режим длительного удержания телеметрии	
Название в макрокоманде: нет	
Параметры:	
`user_id`	Идентификатор пользователя, от имени которого будет включено удержание телеметрии
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.
`allow_display_user`	Признак отображения имени пользователя, захватившего телеметрию (заданного в параметре **user_id**). Возможные значения: • `true` — имя отображается; • `false` — имя не отображается.

4.3.1.2.7.4 TELEMETRY_RELEASE

Описание: Отключить режим длительного удержания телеметрии	
Название в макрокоманде: нет	
Параметры:	
`user_id`	Идентификатор пользователя, от имени которого будет отключено удержание телеметрии
`priority`	Приоритет управления PTZ для данного пользователя. Возможные значения: [`1; 10`]. Значение по умолчанию — `1`.

4.3.1.2.8 Управление дополнительными устройствами камеры

В разделе описаны следующие команды управления дополнительными устройствами камеры:

- **LIGHT_OFF**.
- **LIGHT_ON**.

- **WIPER**.
- **WASHING**.

4.3.1.2.8.1 LIGHT_OFF

Описание: Отключить встроенное освещение камеры
Внимание! Команда обрабатывается в случае, если в настройках *Камеры* параметр **Освещение** принимает значение `Встроенное`.
Название в макрокоманде: Выключить освещение
Параметры: нет

4.3.1.2.8.2 LIGHT_ON

Описание: Включить встроенное освещение камеры
Внимание! Команда обрабатывается в случае, если в настройках *Камеры* параметр **Освещение** принимает значение `Встроенное`.
Название в макрокоманде: Включить освещение
Параметры: нет

4.3.1.2.8.3 WIPER

Описание: Включить дворник
Название в макрокоманде: нет
Параметры: нет

4.3.1.2.8.4 WASHING

Описание: Запустить процедуру мойки камеры
Название в макрокоманде: нет
Параметры: нет

4.3.1.2.9 Команды для постановки задач переноса архива

Внимание! Некоторые команды отправляются объекту с `objectType==VIDEO`. Такие команды отмечены соответствующим элементами оформления.

Примечание. При обработке команд с указанием времени, его следует отправлять в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`.

В разделе описаны следующие команды для постановки задач переноса архива:

- **ARCH_FILES_MOVE_INTERVAL**.
- **ARCH_FILES_ERASE_INTERVAL**.
- **ARCH_FILES_MOVE_INTERVAL (VIDEO)**.

4.3.1.2.9.1 ARCH_FILES_MOVE_INTERVAL

Описание: Добавить команду на перенос оперативного архива в очередь команд

Примечание. Архив в указанном интервале переносится с одного сервера на другой. При успешном выполнении все файлы будут скопированы на сервер-получатель и удалены с сервера-источника. В случае ошибки операция может прерваться в любой момент, но гарантируется дублирование не более чем одного файла на обоих серверах.

Название в макрокоманде: нет

Параметры:

`direction (опциональный)`	Направление переноса архива. Возможные значения: • `from` — архив переносится с сервера, указанного в параметре `host` (см. ниже), на сервер, который получил команду. Является значением по умолчанию. • `to` — архив переносится с сервера, получившего команду, на сервер, указанный в параметре `host`.
`host`	Название хоста, с которого/на который осуществляется перенос оперативного архива. Если перенос выполняется для *Камеры Удаленной системы*, должно включать имя *Удаленной системы*, например, `[RS1#HOST]`.
`time_from`	Дата и время начала временного интервала для переноса архива (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`), задаваемые оператором. **Примечание.** Фактическое время начала временного интервала переноса может отличаться от заданного, т.к. архив переносится по файлам, время начала которых наиболее соответствует заданному значению.
`time_until`	Дата и время окончания временного интервала для переноса архива (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`), задаваемые оператором. **Примечание.** Фактическое время окончания временного интервала переноса может отличаться от заданного, т.к. архив переносится по файлам, время окончания которых наиболее соответствует заданному значению.
`__user (опциональный)`	Идентификатор клиента/оператора, отправившего команду.

4.3.1.2.9.2 ARCH_FILES_ERASE_INTERVAL

Описание: Удалить локальный архив с *Видеосервера*

Примечание. Удаляются файлы оперативного архива в интервале от `time_from` до `time_until` и очищается индекс от них. Успешное выполнение операции подразумевает полное удаление всех файлов за указанный интервал времени.

Название в макрокоманде: нет

Параметры:

`time_from`	Дата и время начала временного интервала для переноса архива (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`), задаваемые оператором **Примечание.** Фактическое время начала временного интервала переноса может отличаться от заданного, т.к. архив переносится по файлам, время начала которых наиболее соответствует заданному значению.
`time_until`	Дата и время окончания временного интервала для переноса архива (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`), задаваемые оператором **Примечание.** Фактическое время окончания временного интервала переноса может отличаться от заданного, т.к. архив переносится по файлам, время окончания которых наиболее соответствует заданному значению.
`__user (опциональный)`	Идентификатор клиента/оператора, отправившего команду

4.3.1.2.9.3 ARCH_FILES_MOVE_INTERVAL (VIDEO)

Внимание! В отличие от описанной выше одноименной команды, отправляемой объекту *Камера* (`objectType==CAM`), данная команда отправляется *Медиасерверу* SecurOS (`objectType==VIDEO`).

Описание: Эквивалентна команде **ARCH_FILES_MOVE_INTERVAL** (для объекта с `objectType==CAM`) по каждой из *Камер*, которая присутствует в списке параметра `cams`

Название в макрокоманде: нет

Параметры:

direction (опциональный)	Направление переноса архива. Возможные значения: • from — архив переносится с сервера, указанного в параметре host (см. ниже), на сервер, который получил команду. Является значением по умолчанию. • to — архив переносится с сервера, получившего команду, на сервер, указанный в параметре host.
cams	Список строковых идентификаторов *Камер*, разделенных запятой. Если перенос выполняется для *Камер Удаленной системы*, идентификаторы *Камер* должны включать имя *Удаленной системы*, например, [RS1#1, RS1#3, RS1#4].
host	Название хоста, с которого/на который осуществляется перенос оперативного архива. Если перенос выполняется для *Камеры Удаленной системы*, должно включать имя *Удаленной системы*, например, [RS1#HOST].
time_from	Дата и время начала временного интервала для переноса архива (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX), задаваемые оператором. **Примечание.** Фактическое время начала временного интервала переноса может отличаться от заданного, т.к. архив переносится по файлам, время начала которых наиболее соответствует заданному значению.
time_until	Дата и время окончания временного интервала для переноса архива (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX), задаваемые оператором. **Примечание.** Фактическое время окончания временного интервала переноса может отличаться от заданного, т.к. архив переносится по файлам, время окончания которых наиболее соответствует заданному значению.
__user (опциональный)	Идентификатор клиента/оператора, отправившего команду.

4.3.1.2.10 Команды управления задачами переноса архива

Примечание. При обработке команд с указанием времени, его следует отправлять в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX.

В разделе описаны следующие команды управления задачами переноса архива:

- **ARCH_FILES_GET_OP_LIST**.
- **ARCH_FILES_OP_CANCEL**.
- **ARCH_FILES_CLEAR_FINISHED_OP**.

Внимание! Все описанные команды отправляются *Медиасерверу* SecurOS (`objectType==VIDEO`).

4.3.1.2.10.1 ARCH_FILES_GET_OP_LIST

Внимание! В отличие от команд, отправляемых объекту *Камера* (`objectType==CAM`), данная команда отправляется *Медиасерверу* SecurOS (`objectType==VIDEO`).

Описание: Запросить список задач переноса архива

Примечание. В ответе возвращаются задачи, находящиеся в процессе выполнения, в очереди или выполненные недавно (история автоматически очищается) на том *Видеосервере*, которому отправлена команда.

Название в макрокоманде: нет

Параметры:

`time_from (опциональный)`	Дата и время, после которого должна начаться операция, чтобы попасть в список задач (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`)
`type_filter (опциональный)`	Фильтр по типу задач — список строковых идентификаторов, разделенных запятыми. Возможные значения идентификаторов: • `move` — задачи перемещения архива; • `erase` — задачи удаления перемещенного архива.

Ответ отправляется в сообщении `VIDEO|slaveId|ARCH_FILES_OP_LIST`, отправляемом как в случае ошибки выполнения команды, так и в случае успешного выполнения, где `slaveId` — идентификатор *Видеосервера*, которому отправлена команда. В сообщении выводятся следующие параметры:

- `time` — время актуальности данных ответа. В случае ошибки не заполняется.
- `operations` — структура в формате `JSON`, содержащая список задач. Описание параметров структуры приведено в Таблице 13.
- `error` — код ошибки. Заполняется только в случае ошибки выполнения команды.
- `message` — текстовое описание кода ошибки. Заполняется только в случае ошибки выполнения команды.

Таблица 13. Параметры `JSON`-структуры `operations`

Параметр	Описание
Блок информации о задаче	
`op_id`	Идентификатор задачи

op_type	Тип задачи. Возможные значения: • move — задача перемещения архива; • erase — задача удаления архива.
op_scheduled	Время постановки задачи в очередь
op_started	Время начала выполнения задачи
op_finished	Время окончания выполнения задачи
op_user	Имя пользователя, создавшего задачу
Блок информации о запросе	
cam	Идентификатор *Камеры*, архив которой переносится
time_from	Дата и время начала временного интервала архива, заданные оператором
time_until	Дата и время окончания временного интервала архива, заданные пользователем
Блок информации о результате	
state	Текущее состояние задачи. Может принимать следующие значения: • scheduled — задача помещена в очередь; • started — задача начала выполняться; • finished — задача завершилась; • cancelled — задача отменена оператором; • failed — задача завершилась с ошибкой.
progress	Прогресс выполнения задачи (в процентах)

4.3.1.2.10.2 ARCH_FILES_OP_CANCEL

Внимание! В отличие от команд, отправляемых объекту *Камера* (objectType==CAM), данная команда отправляется *Медиасерверу* SecurOS (objectType==VIDEO).

Описание: Отменить задачу с указанным идентификатором

Примечание. Отменить можно любую задачу, поставленную любым *Пользователем* SecurOS на любом *Видеосервере*. Если задача была отменена до того, как успела завершиться, то по ней будет послано событие **ARCH_FILES_OP_CANCELLED**.

Название в макрокоманде: нет

Параметры:

op_id	Идентификатор отменяемой задачи

4.3.1.2.10.3 ARCH_FILES_CLEAR_FINISHED_OP

Внимание! В отличие от команд, отправляемых объекту *Камера* (objectType==CAM), данная команда отправляется *Медиасерверу* SecurOS (objectType==VIDEO).

Описание: Удалить завершенные задачи с указанными идентификаторами из списка задач

Примечание. Удалить можно любую задачу, поставленную любым *Пользователем* SecurOS на любом *Видеосервере*. В случае ошибки возвращает ответ, содержащий список **op_id**, по которым не удалось удалить операции (задача еще не завершилась, идентификатор задачи указан неверно или задача с указанным идентификатором отсутствует).

Название в макрокоманде: нет

Параметры:

op_ids (опциональный)	Список идентификаторов задач, подлежащих удалению из списка завершенных. Если не задан или отсутствует, удаляются все завершенные задачи

4.3.1.3 Работа с субтитрами

Текст, который накладывается на отображаемое видео, называется субтитрами. В SecurOS субтитры выводятся в ячейке *Камеры* в области, лежащей внутри кадра. Размеры этой области ограничены полями 10% (сверху/снизу) и 15% (слева/справа) от размера кадра. Значения полей фиксированы и не могут быть изменены пользователем. Вывести субтитры за пределами этой области невозможно.

Для работы с субтитрами предназначены две команды:

- **ADD_SUBTITLES** — добавление субтитров в видеопоток (см. **Добавление субтитров**);
- **CLEAR_SUBTITLES** — очистка видеопотока от субтитров (см. **Удаление субтитров**).

Добавление субтитров

Внимание! Для отображения субтитров в режимах **Он-лайн/Архив** отметьте флажок **Отображать субтитры** в настройках объекта *Медиа Клиент* (см. **Руководство администратора SecurOS**). В противном случае субтитры выводится не будут.

В Листинге 10 приведен пример команды добавления субтитров при распознавании в кадре автомобиля с номером из черного списка. Субтитры выводятся для каждого события распознавания в одной и той же позиции, для чего выполняется автоматическая очистка старого значения.

Листинг 10. Добавление субтитров

```
const sos=require('securos');
sos.connect(async function(core)
{
      core.registerEventHandler("LPR_LOGIC", "1", "CAR_LP_FOUND", AddSub);
```

```
      function AddSub(e)
      {
            if(e.params["database_type"] == "blacklist")
            {
                  core.doReact("CAM", "1", "CLEAR_SUBTITLES");
                  core.doReact("CAM", "1", "ADD_SUBTITLES", {"command":
                              "<p style = margin-left:320px;color:red;
                              >Черный список "+ e.number +"<br></p>"});
            }
      }
});
```

Набор символов, доступный для использования в тексте субтитров, определяется системным параметром Windows **Язык программ, не поддерживающих Юникод** (**Панель управления → Часы, язык и регион → Язык и региональные стандарты → Закладка Дополнительно**). Символы **Основной латиницы** (ASCII) доступны всегда. Внутри параметра `command` возможны следующие действия со строкой субтитров:

- **Форматирование текста**;
- **Позиционирование строки**;
- **Использование Unicode**.

Форматирование текста

Для форматирования текста могут использоваться любые атрибуты HTML тега `<font>` (`color`, `face`, `size`).

> **Внимание!** Значения атрибутов `face` и `size` влияют на количество символов в строке.

Позиционирование строки

В ячейке *Камеры* субтитры выводятся в терминальном режиме, т.е. область вывода заполняется строками текста сверху-вниз. При достижении нижней границы области вывода верхние строки замещаются новыми, которые выводятся снизу. Если в команде не определено иначе, текст субтитра выводится в терминальную строку посимвольно, в зависимости от значения параметров **Минимальный размер шрифта** и **Символов в строке** (см. **Руководство администратора SecurOS**, настройки **Медиа Клиента**, закладка **Камеры**).

По умолчанию, начало строки субтитров расположено в левом верхнем углу области вывода. Это значение является фиксированным и не может быть изменено пользователем. Смещение начала строки относительно начальной точки можно задать следующим образом:

- по горизонтали — с помощью CSS-атрибутов `margin-left/margin-right` или HTML-атрибута `align` в тегах `<p>` или `<div>`. Например:

```
<p style = margin-left:150px;color:cyan>Тестовая строка субтитров</p>
```

- по вертикали — выравнивание строки субтитров по вертикали не предусмотрено. При необходимости можно использовать принудительный перенос строки с помощью элемента `<br>`.

Использование Unicode
Специальные и прочие символы наиболее полно представлены в Unicode-таблицах символов. Для добавления любого Unicode-символа в текст субтитра используйте десятичное (`&#DDDD;`) или шестнадцатиричное (`&#xHHHH;`) значение кода символа, либо его мнемоническое представление (`&entity`). Например, чтобы добавить в текст субтитра символ евро, используйте, соответственно, значение `"&#8364;"`, `"&#x20AC"` или `"&euro"`. Коды Unicode-символов можно посмотреть на странице https://unicode-table.com.

Удаление субтитров
Субтитры можно удалять автоматически (см. Листинг 10), либо вручную, в любой момент, когда это требуется оператору.

Простейшим способом удаления субтитров в ручном режиме является запуск пустой макрокоманды. Пример использования команды удаления субтитров приведен в Листинге 11.

Листинг 11. Удаление субтитров

```
const sos=require('securos');
sos.connect(async function(core)
{
      core.registerEventHandler("MACRO", "1.1", "RUN", () =>
      {
            core.doReact("CAM", "1", "CLEAR_SUBTITLES");
      });
});
```

Для корректной работы скрипта создайте объект *Макрокоманда* (без параметров). Кадр будет очищаться от субтитров каждый раз, когда оператор нажмет кнопку этой *Макрокоманды* на *Панели управления*.

4.3.2 Зона

Идентификатор типа объекта: `CAM_ZONE`.

Таблица 14. События CAM_ZONE

Идентификатор события	Название в макрокоманде	Описание
`ARM`	Поставлена на охрану	
`DISARM`	Снята с охраны	
`MD_START`	Тревога	
`MD_STOP`	Конец тревоги	

Таблица 15. Действия CAM_ZONE

Идентификатор действия	Название на карте/в макрокоманде	Описание
`ARM`	Поставить на охрану	

Идентификатор действия	Название на карте/в макрокоманде	Описание
DISARM	Снять с охраны	

4.3.3 Световой детектор

Идентификатор типа объекта: LD.

Таблица 16. События LD

Идентификатор события	Название в макрокоманде	Описание
LIGHT_ON	Свет вкл.	
LIGHT_OFF	Свет выкл.	

Действия: отсутствуют.

4.3.4 Конвертер архива

Идентификатор типа объекта: ARCH_CNV.

Таблица 17. События ARCH_CNV

Идентификатор события: TASK_STARTED	
Название в макрокоманде: Начата конвертация данных	
Описание: Задача поставлена на исполнение	
Параметры:	
task_id	Идентификатор задачи, присвоенный при постановке в очередь
session_magic	Переданный в запросе идентификатор сессии
Идентификатор события: TASK_FINISHED	
Название в макрокоманде: Завершена конвертация данных	
Описание: Задача успешно выполнена	
Параметры:	
task_id	Идентификатор задачи, присвоенный при постановке в очередь
session_magic	Переданный в запросе идентификатор сессии
output_files	Список созданных файлов с указанием полного пути. Имена файлов перечисляются через точку с запятой (;)

actual_start_time	Дата и время фактического начала видео в экспортированном файле (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX) **Примечание.** При конвертации архивов нескольких камер в формат *Evidence* значение не выводится.
actual_stop_time	Дата и время фактического окончания видео в экспортированном файле (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX) **Примечание.** При конвертации архивов нескольких камер в формат *Evidence* значение не выводится.

Идентификатор события: TASK_INCOMPLETE

Название в макрокоманде: Задача выполнена частично

Описание: Событие генерируется в следующих случаях:

1. При конвертации архива нескольких камер в формат *Evidence* архивы некоторых камер экспортировать не удалось.
2. Не удалось получить информацию о *Долговременном архиве*.

Параметры:

task_id	Идентификатор задачи, присвоенный при постановке в очередь
session_magic	Переданный в запросе идентификатор сессии
output_files	Имя созданного файла с указанием полного пути
comment	В зависимости от причин генерации (см. **Описание**) может содержать: 1. Информацию о *Камерах*, экспорт архива которых завершен с ошибкой. 2. Сообщение Долговременный архив недоступен.

Идентификатор события: TASK_CANCELLED

Название в макрокоманде: Конвертация данных отменена

Описание: Задача конвертации архива отменена оператором

Параметры:

session_magic	Переданный в запросе идентификатор сессии
operator_name	Имя оператора, отменившего задачу
operator	Идентификатор оператора, отменившего задачу

Идентификатор события: TASK_FAILED

Название в макрокоманде: Ошибка конвертации данных

Описание: В ходе выполнения задачи возникли проблемы	
Параметры:	
`task_id`	Идентификатор задачи, присвоенный при постановке в очередь
`problem`	Текстовое описание возникшей проблемы
`session_magic`	Переданный в запросе идентификатор сессии

Таблица 18. Действия `ARCH_CNV`

Идентификатор действия: `ARCH_EXPORT`	
Название в макрокоманде: нет	
Описание: Экспорт архива	
Параметры:	
`channel_id` (обязательный)	Идентификатор запрашиваемого канала. В качестве значения необходимо указать идентификатор камеры, для которой должна производиться конвертация архива
`time_start` (обязательный)	Левая граница запрашиваемого временного интервала (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`)
`time_end` (обязательный)	Правая граница запрашиваемого временного интервала (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`)
`mic_id`	Идентификатор свободного микрофона при конвертации только аудио. Если присутствует в команде, параметр `channel_id` игнорируется
`session_magic` (опциональный)	Идентификатор запроса, будет включен в ответ конвертера
`operator` (опциональный)	Имя пользователя (оператора), который запускает процесс конвертации
`comment` (опциональный)	Комментарий к запускаемому процессу конвертации
`client_type` (опциональный)	Тип объекта, от которого пришел запрос на конвертацию. Значение параметра используется для адресации уведомлений о результатах конвертации. Значение параметра будет передаваться в ответном уведомлении без изменений (при его наличии)
`client_id` (опциональный)	Идентификатор объекта, от которого пришел запрос на конвертацию. Значение параметра используется для адресации уведомлений о результатах конвертации. Значение параметра будет передаваться в ответном уведомлении без изменений (при его наличии)

`dir (опциональный)`	Директория, в которую будет записан сконвертированный файл (директория экспорта). В качестве значения необходимо указать значение директории в абсолютном формате. Если директория не задана, будет использоваться значение директории из настроек *Конвертера архива*. В шаблоне имени директории допускается использование макроподстановок (см. **Руководство администратора SecurOS**)
`file_name (опциональный)`	Имя файла, который будет создан в результате конвертации. Всегда задается без указания пути. Если имя файла не задано, будет использоваться имя из настроек *Конвертера архива*. В шаблоне имени файла допускается использование макроподстановок (см. **Руководство администратора SecurOS**)
`arch_id (опциональный)`	Имя *Компьютера*, дочерним к которому является *Архиватор*. Используется для конвертации из долговременного архива. Задается в формате `<Имя Компьютера.ID Архиватора>`. Если значение не задано, конвертация возможна только для оперативного архива
`arch_port (опциональный)`	Порт связи с *Архиватором*. Возможное значение — `901`. Если значение не задано или некорректно, конвертация возможна только для оперативного архива
`password (опциональный)`	Используется для шифрования файла при конвертации в формат *Evidence*. Если параметр задан и не является пустым, файл будет зашифрован с применением алгоритма, выбранного в настройках объекта *Конвертер архива*, и защищен указанным паролем. В противном случае шифрование файла не применяется. **Внимание!** При задании пароля допускается использовать только символы латинского алфавита и цифры.
`password_not_encoded (опциональный)`	Флаг шифрования пароля при передаче по сети. Возможные значения: • `0` (значение по умолчанию) или не задан — пароль передается в зашифрованном виде. • `1` — пароль передается в открытом виде. **Внимание!** Если процедура экспорта инициируется из скрипта, должен принимать значение `1`.

4.3.5 Архиватор

Идентификатор типа объекта: `ARCHIVER`.

Таблица 19. События `ARCHIVER`

Идентификатор события: `TASK_ADDED`

Название в макрокоманде: Задача добавлена в очередь	
Описание: Задача копирования архива поставлена в очередь задач	
Параметры:	
`task_id`	Идентификатор задачи, присвоенный при постановке в очередь
`operator`	Идентификатор оператора, от лица которого была запущена команда. Необязательный параметр. Присутствует, если был задан при постановке задачи в очередь
Идентификатор события: `ADD_TASK_FAILED`	
Название в макрокоманде: Ошибка добавления задачи в очередь	
Описание: При постановке задачи копирования в очередь возникла ошибка	
Параметры:	
`task_id`	Идентификатор задачи, присвоенный при постановке в очередь
`operator`	Идентификатор оператора, от лица которого была запущена команда. Необязательный параметр. Присутствует, если был задан при постановке задачи в очередь
Идентификатор события: `TASK_STARTED`	
Название в макрокоманде: Архивация начата	
Описание: Началось выполнение задачи копирования	
Параметры:	
`task_id`	Идентификатор задачи, присвоенный при постановке в очередь
`operator`	Идентификатор оператора, от лица которого была запущена команда. Необязательный параметр. Присутствует, если был задан при постановке задачи в очередь
Идентификатор события: `TASK_FINISHED`	
Название в макрокоманде: Архивация завершена	
Описание: Задача успешно выполнена	
Параметры:	
`task_id`	Идентификатор задачи, присвоенный при постановке в очередь
`operator`	Идентификатор оператора, от лица которого была запущена команда. Необязательный параметр. Присутствует, если был задан при постановке задачи в очередь
Идентификатор события: `TASK_FAILED`	
Название в макрокоманде: Ошибка архивации	

Описание: В ходе выполнения задачи возникли проблемы	
Параметры:	
`task_id`	Идентификатор задачи, присвоенный при постановке в очередь
`operator`	Идентификатор оператора, от лица которого была запущена команда. Необязательный параметр. Присутствует, если был задан при постановке задачи в очередь
Идентификатор события: `TASK_CANCELLED`	
Название в макрокоманде: Архивация отменена	
Описание: Задача копирования архива отменена	
Параметры:	
`task_id`	Идентификатор задачи, присвоенный при постановке в очередь
`operator`	Идентификатор оператора, от лица которого была запущена команда. Необязательный параметр. Присутствует, если был задан при постановке задачи в очередь
Идентификатор события: `TASK_NOT_FOUND`	
Название в макрокоманде: Задача не найдена	
Описание: Указанная задача отсутствует в очереди задач	
Параметры:	
`task_id`	Идентификатор задачи, присвоенный при постановке в очередь. Необязательный параметр. Присутствует в событии, если выполнялась команда `CANCEL_TASK`
`cam_id`	Идентификатор *Камеры*. Необязательный параметр. Присутствует в событии, если выполнялась команда `CANCEL_CAM_TASKS`
`operator`	Идентификатор оператора, от лица которого была запущена команда. Необязательный параметр. Присутствует, если был задан при постановке задачи в очередь

Таблица 20. Действия `ARCHIVER`

Идентификатор действия: `ADD_TASK`	
Название в макрокоманде: Добавить задачу	
Описание: Добавить новую задачу копирования архива в очередь задач	
Параметры:	
`cam_id`	Идентификатор *Камеры*, архив которой необходимо копировать
`begin_time`	Дата и время начала копируемого фрагмента (в формате `ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX`)

end_time	Дата и время окончания копируемого фрагмента (в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX). Необязательный параметр. Если не указан, за время окончания фрагмента принимается время отправки команды
task_id	Идентификатор задачи, уникальный в рамках сессии *Архиватора*. Обязательный параметр. Значение должно быть задано: • в параметрах объекта *Макрокоманда* — вручную; • в параметрах объекта *Скрипт Node.js* — с помощью произвольного алгоритма (последовательность, генератор).
operator	Идентификатор оператора, от лица которого была запущена команда. Необязательный параметр
Идентификатор действия: CANCEL_TASK	
Название в макрокоманде: Отменить задачу	
Описание: Остановить копирование и удалить задачу из очереди	
Параметры:	
task_id	Идентификатор задачи, которую необходимо отменить
Идентификатор действия: CANCEL_CAM_TASKS	
Название в макрокоманде: Отменить все задачи Камеры	
Описание: Остановить копирование и удалить из очереди все задачи копирования архива указанной камеры	
Параметры:	
cam_id	Идентификатор *Камеры*, для которой необходимо отменить все задачи копирования архива
Идентификатор действия: CANCEL_ALL_TASKS	
Название в макрокоманде: Отменить все задачи	
Описание: Остановить копирование и удалить из очереди все задачи данного *Архиватора*	
Параметры: нет	

4.3.6 Медиа Клиент

Идентификатор типа объекта: MEDIA_CLIENT.

Таблица 21. События MEDIA_CLIENT

Идентификатор события: STATE
Название в макрокоманде: нет

Описание: Содержит описание текущего состояния *Медиа Клиента* в формате JSON	
Параметры:	
`__to`	Имя клиента, которому будет отправлено сообщение. Опциональный параметр. Присутствует в сообщении только в том случае, если в запросе **GET_STATE** был указан параметр `__from`. В противном случае осуществляется широковещательная рассылка сообщения.
`state`	Текущее состояние *Медиа Клиента* в формате JSON. Содержит следующую информацию: • Текущая раскладка *Медиа Клиента* (видео/аудио) и/или используемый *Вид*; • *Камеры*, отображаемые в настоящий момент, и номера их ячеек; • Режим работы *Камеры/Микрофона* (живое видео/архив); • Активная *Камера/Микрофон*.

Таблица 22. Действия `MEDIA_CLIENT`

Идентификатор действия: `SEEK`	
Название в макрокоманде: нет	
Описание: Позиционировать архив на указанном времени	
Параметры:	
`date`	Дата (в формате `ДД-ММ-ГГ`).
`time`	Время (в формате `ЧЧ:ММ:СС.XXX`).
`cam (опциональный)`	Идентификатор камеры, которую необходимо перевести в режим архива до выполнения операции позиционирования. Если не задан, количество камер, находящихся в режиме архива, не изменится
Идентификатор действия: `CAM_MODE` **Внимание!** Если *Медиа Клиент* находится в режиме `Тревожный` или `Только онлайн` (см. **Руководство администратора SecurOS**), команда игнорируется.	
Название в макрокоманде: нет	
Описание: Установить режим работы камеры	
Параметры:	
`cams`	Идентификаторы камер, разделенные запятой.

mode	Режим работы камеры. Возможные значения: • live — работа в режиме живого видео; • arch — работа в режиме архива. **Внимание!** Параметр влияет только на камеры, расположенные на текущей видеостранице *Медиа Клиента*. Все прочие камеры данного *Медиа Клиента*, находящиеся в режиме архива, переводятся в режим живого видео.

Идентификатор действия: ACTIVATE_CAM

Название в макрокоманде: нет

Описание: Активировать камеру

Параметры:

cam	Идентификатор камеры, которую необходимо активировать. При выполнении команды отображается видеостраница, на которой расположена активированная камера. Текущая раскладка *Медиа Клиента* при этом не изменяется

Идентификатор действия: ACTIVATE_NAMED_MARKUP

Примечание. Команда будет проигнорирована, если выполняется одно из следующих условий:
1. Работа с *Видами* запрещена в настройках данного *Медиа Клиента*.
2. Работа с заданным *Видом* запрещена в настройках данного *Медиа Клиента* или заданный *Вид* не существует.
3. Работа со всеми *Камерами* заданного *Вида* запрещена в настройках данного *Медиа Клиента*.
4. На момент получения команды *Медиа Клиентом* в нем открыт редактор *Видов*.

Название в макрокоманде: нет

Описание: Активировать *Вид*

Параметры:

id	Идентификатор *Вида*, который необходимо активировать на *Медиа Клиенте*. **Примечание.** Если задано значение "*" (звездочка), будет активирован *Вид* **Все устройства**.

Идентификатор действия: HARD_FILTER

Название в макрокоманде: нет

Описание: Задать фильтр, в котором перечисляются *Камеры*, отображение которых на *Медиа Клиент* разрешено/запрещено. Такой фильтр является дополнительным и применяется к текущим настройкам *Медиа Клиента*, относящимся к работе с *Камерами*

Параметры:

`cams`	Список идентификаторов камер, разделенных символом "`,`" (запятая). При выполнении команды действуют следующие правила: • Отображаются/скрываются все *Камеры* из списка, если их идентификаторы существуют и эти *Камеры* выбраны в настройках *Медиа Клиента*; • Если задан несуществующий идентификатор, который является единственным элементом списка: – не отображается ни одна *Камера*, если параметр `forbid = 0`; – отображаются все *Камеры*, если параметр `forbid = 1`. • Если список пуст, отображаются все *Камеры*, выбранные в текущих настройках *Медиа Клиента*.
`forbid`	Флаг разрешения/запрета отображения камер, перечисленных в параметре `cams`. Возможные значения: • `0` — отображение перечисленных *Камер* на *Медиа Клиенте* разрешено; • `1` — отображение перечисленных *Камер* на *Медиа Клиенте* запрещено.

Идентификатор действия: `PLAY`

Название в макрокоманде: нет

Описание: Начать воспроизведение архива

Параметры: нет

Идентификатор действия: `STOP`

Название в макрокоманде: нет

Описание: Остановить воспроизведение архива

Параметры: нет

Идентификатор действия: `NEXT_FRAME`

Название в макрокоманде: нет

Описание: Позиционировать указанную камеру на следующий кадр архива

Параметры:

`cam`	Идентификатор камеры

Идентификатор действия: `PREV_FRAME`

Название в макрокоманде: нет

Описание: Позиционировать указанную камеру на предыдущий кадр архива

Параметры:	
`cam`	Идентификатор камеры
Идентификатор действия: `NEXT_RECORD`	
Название в макрокоманде: нет	
Описание: Позиционировать указанную камеру на следующую запись архива	
Параметры:	
`cam`	Идентификатор камеры
Идентификатор действия: `PREV_RECORD`	
Название в макрокоманде: нет	
Описание: Позиционировать указанную камеру на предыдущую запись архива	
Параметры:	
`cam`	Идентификатор камеры
Идентификатор действия: `ADD_SEQUENCE`	
Название в макрокоманде: нет	
Описание: Установить раскладку *Медиа Клиента* (с указанием камер)	
Внимание! Если *Медиа Клиент* находится в режиме `Тревожный` или `Активный` (см. **Руководство администратора SecurOS**), команда игнорируется. В режиме `Только он-лайн` выполняется переключение раскладки, при этом все камеры остаются в режиме отображения живого видео.	
Параметры:	
`mode`	Идентификатор раскладки. Возможные значения: • идентификаторы стандартных и широкоформатных раскладок: – `"1x1"` — раскладка `1x1`; – `"2x2"` — раскладка `2x2`; – `"1w5"` — раскладка `1+5`; – `"4w3"` — раскладка `4+3` и т.д. • идентификатор пользовательского объекта *Раскладка*.
`cams_per_page`	Количество камер, выводимых на одной видеостранице *Медиа Клиента*.
Внимание! Если параметры `mode` и `cams_per_page` не заданы, раскладка выбирается системой автоматически исходя из количества камер, указанных в параметре `seq`.	

`seq`	Список идентификаторов камер, разделенных символом "`|`". Позиция в списке соответствует номеру ячейки, в которую будет выведена камера. **Примечание.** Камеры размещаются в ячейках слева направо сверху вниз. Если список содержит пустые, некорректно заданные или не существующие в системе идентификаторы камер, ячейки, соответствующие позициям таких идентификаторов, остаются пустыми.
`arch`	Список идентификаторов камер в архивном режиме, разделенных символом "`,`" (запятая). При выполнении команды учитываются только камеры, перечисленные в предыдущем списке. **Внимание!** Все прочие камеры данного *Медиа Клиента*, находящиеся в режиме архива, переводятся в режим живого видео.

Идентификатор действия: `SET_MARKUP`

Название в макрокоманде: нет

Описание: Установить раскладку *Медиа Клиента* (без указания камер)

Внимание! Если *Медиа Клиент* находится в режиме `Тревожный` или `Активный` (см. **Руководство администратора SecurOS**), команда игнорируется.

Параметры:

`mode`	Идентификатор раскладки. Возможные значения: • идентификаторы стандартных и широкоформатных раскладок: – "`1x1`" — раскладка `1x1`; – "`2x2`" — раскладка `2x2`; – "`1w5`" — раскладка `1+5`; – "`4w3`" — раскладка `4+3` и т.д. • идентификатор пользовательского объекта *Раскладка*.

Идентификатор действия: `EXPORT_FRAME_FROM_ACTIVE_CAM`

Название в макрокоманде: нет

Описание: Экспорт кадра с активной камеры в файл

Примечание. Экспортируется кадр, отображаемый в ячейке камеры на момент выполнения команды.

Параметры: нет

Идентификатор действия: `ADD_BOOKMARK_TO_ACTIVE_CAM`

Название в макрокоманде: нет

Описание: Установить закладку на отображаемый кадр активной камеры. Команда доступна при записи живого видео и при просмотре архива.

Примечание. Закладка устанавливается на кадр, отображаемый в ячейке камеры на момент выполнения команды.

Параметры: нет

Идентификатор действия: `JOYSTICK`

Название в макрокоманде: нет

Описание: Управление PTZ активной камеры

Параметры:

`pan`	Панорамирование
`tilt`	Наклон
`zoom`	Масштаб

Идентификатор действия: `MAXIMIZE_OR_RESTORE`

Название в макрокоманде: нет

Описание: Управление размером ячейки активной камеры: развернуть во весь экран/свернуть до прежнего размера

Параметры: нет

Идентификатор действия: `SWITCH_PAGE_TO_ARCHIVE`

Название в макрокоманде: нет

Описание: Перевести все камеры текущей видеостраницы в архив

Параметры: нет

Идентификатор действия: `SWITCH_ALL_CAMERAS_TO_LIVE`

Название в макрокоманде: нет

Описание: Вывести все камеры из архива

Параметры: нет

Идентификатор действия: `KEY_PRESSED`

Название в макрокоманде: нет

Описание: Команды управления *Медиа Клиентом* и объектами

Примечание. Команды управления камерой применяются к активной камере.

Параметры:

`key`	Возможные значения: • `ARM` — поставить камеру на охрану; • `DISARM` — снять камеру с охраны; • `REC` — включить запись; • `REC_STOP` — выключить запись; • `LEFT` — влево; • `RIGHT` — направо; • `UP` — вверх; • `DOWN` — вниз; • `NEXT_PAGE` — перейти на следующую видеостраницу; • `PREV_PAGE` — перейти на предыдущую видеостраницу; • `FF` — перейти к следующему кадру архива; • `REW` — перейти к предыдущему кадру архива; • `PLAY` — начать воспроизведение архивной записи; • `STOP` — остановить воспроизведение архивной записи; • `MODE_ARCH` — перевести камеру в режим архива; • `MODE_VIDEO` — перевести камеру в режим просмотра живого видео.

Идентификатор действия: `LAYOUT_AUTOSCROLL`

Название в макрокоманде: нет

Описание: Начать/остановить автолистание *Видов* или видеостраниц *Вида* `Все объекты`. Листание осуществляется по *Видам*, размещенным на *Панели быстрого доступа к Видам*. Если на *Панели быстрого доступа* размещено меньше двух *Видов*, листание осуществляется по всем *Видам* из *Списка видов*. Если на момент отправки команды активен *Вид* `Все объекты`, выполняется автолистание страниц этого *Вида*.

Параметры:

`enable`	Возможные значения: • `0` — остановить автолистание; • `1` — начать автолистание.

Идентификатор действия: `PLACE_CAMERA`

Название в макрокоманде: нет

Описание: Переместить *Камеру*/*Камеры* в указанную ячейку(-и) раскладки

Параметры:	
`cam`	Идентификатор *Камеры* или список идентификаторов *Камер*, которые надо переместить. При передаче списка значений в качестве разделителя используется символ `"|"` (вертикальная черта), например, `"14|59|62"`. **Внимание!** Если значение не задано, *Камера* удаляется из указанной ячейки раскладки.
`cell`	Номер ячейки (в текущей раскладке), куда необходимо переместить *Камеру*
`autolocate`	Включение/выключение режима автозаполнения ячеек. Возможные значения: • `0` — номер ячейки для перемещения *Камеры* определяется параметром `cell`; • `1` — режим автозаполнения. Номер ячейки для перемещения *Камеры*, заданный параметром `cell`, игнорируется. Если выбран режим автозаполнения (значение `1`), ячейки *Рабочей области Медиа Клиента* будут заполняться следующим образом: • Изменяется только текущая страница *Медиа Клиента*. • Если на *Медиа Клиенте* есть активная *Камера*, то камера с указанным идентификатором будет перемещена в ее ячейку. При перемещении списка *Камер* в ячейку будет помещена последняя *Камера* из списка, указанного в параметре `cam`. • Если поступила команда на перемещение *Камеры*, которая уже имеется на данной раскладке, она игнорируется. *Камера* остается в текущей ячейке. • В остальных случаях, включая перемещение списка *Камер*, заполнение происходит слева-направо и сверху-вниз, независимо от наличия *Камер* в ячейках раскладки. При заполнении раскладки происходит замещение по кругу. При этом, если между командами заполнения ячеек выполнялись прочие команды управления *Медиа Клиентом* (например, активация конкретной камеры и т.д.), то заполнение ячеек начинается заново, с верхней левой ячейки. При перемещении списка *Камер* ячейки будут заполняться по порядку перечисления идентификаторов *Камер* в списке параметра `cam`.

Идентификатор действия: `SWITCH_LAYOUT`

Название в макрокоманде: нет

Описание: Перейти к следующему/предыдущему *Виду*. Переход осуществляется по *Видам*, размещенным на *Панели быстрого доступа к Видам*. Если на *Панели быстрого доступа* размещено меньше двух *Видов*, переход осуществляется по всем *Видам* из *Списка видов*. *Вид* `Все объекты` игнорируется.

Параметры:

relative	Возможные значения: • <=0 — переход к предыдущему *Виду*; • >0 — переход к следующему *Виду*.
Идентификатор действия: TELEMETRY_STATE	
Название в макрокоманде: нет	
Описание: Блокировать управление телеметрией. При поступлении команды анализируется приоритет управления PTZ для пользователя, отправившего команду. Будет выполняться та команда управления, которая отправлена пользователем с наибольшим значением приоритета. Управление PTZ для остальных пользователей будет заблокировано.	
Параметры:	
TELEMETRY_BUSY	Признак блокировки управления телеметрией. Возможные значения: • 0 — управление заблокировано; • 1 — управление разблокировано.
SLAVE_ID	Идентификатор *Компьютера*, с которого была отправлена команда управления телеметрией.
my_obj_slave	Идентификатор *Компьютера*, на котором расположена управляемая *Камера*.
owner_id	Имя *Пользователя*, отправившего команду управления (поле **Название** объекта *Пользователь*).
priority	Приоритет управления PTZ для данного *Пользователя*. Возможные значения: [0; 2147483647]. Значение по умолчанию — 0. **Примечание.** В первую очередь будет выполняться команда с наибольшим значением приоритета.
cam_id	Идентификатор *Камеры*, управляемой данным *Пользователем*.
time	Время отправки команды управления, в формате ЧЧ:MM:CC.XXX.
date	Дата отправки команды, в формате ДД-ММ-ГГ.
Идентификатор действия: GET_STATE	
Название в макрокоманде: нет	
Описание: Получить текущее состояние *Медиа Клиента*	
Параметры:	
__from	Имя внешнего клиента, который отправил запрос на получение текущего состояния *Медиа Клиента*. Опциональный параметр. Если присутствует в команде, то в ответном сообщении **STATE** будет присутствовать параметр __to, содержащий имя клиента.

Пример команды ADD_SEQUENCE приведен ниже.

Листинг 12. Пример команды ADD_SEQUENCE.

```
let params = {
    "mode": "2x2",
    "seq": "1|2|3|4",
    "arch": "2,4"
     };
core.doReact("MEDIA_CLIENT", "1", "ADD_SEQUENCE", params);
/* Установить раскладку 2x2 и вывести камеры с идентификаторами 1, 2,
/* 3, 4 в ячейки с соответствующими позициями, камеры с идентификаторами
/* 2 и 4 вывести в архивном режиме

let params = {
    "mode": "2x2",
    "seq": "1|||2",
     };
core.doReact("MEDIA_CLIENT", "1", "ADD_SEQUENCE", params);
/* Установить раскладку 2x2 и вывести камеры с идентификаторами 1, 2
/* в первую и последнюю ячейки. Ячейки 2 и 3 - пустые
```

4.3.7 Image Processor

Идентификатор типа объекта: IMAGE_EXPORT.

Таблица 23. События IMAGE_EXPORT

Идентификатор события: TASK_QUEUE_OVERLOADED	
Название в макрокоманде: нет	
Описание: Очередь заданий *Image Processor* переполнена. Постановка новых заданий в очередь приостанавливается до разгрузки очереди (см. сообщение TASK_QUEUE_UNDERLOADED ниже)	
Параметры: нет	
Идентификатор события: TASK_QUEUE_UNDERLOADED	
Название в макрокоманде: нет	
Описание: Очередь заданий *Image Processor* разгружена	
Параметры: нет	
Идентификатор события: EXPORT_FAILED	
Название в макрокоманде: нет	
Описание: Ошибка экспорта	
Параметры:	
request_id	Идентификатор, заданный в запросе на экспорт изображения

Идентификатор события: EXPORT_DONE	
Название в макрокоманде: нет	
Описание: Экспорт выполнен успешно	
Параметры:	
request_id	Идентификатор, заданный в запросе на экспорт изображения

Таблица 24. Действия IMAGE_EXPORT

Идентификатор действия: EXPORT	
Название в макрокоманде: нет	
Описание: Обработка и экспорт изображения (кадра) в файл или базу данных. **Примечание.** Пример команды EXPORT приведен в Листинге 13.	
Параметры:	
request_id	Идентификатор запроса. Заданное значение будет передаваться в ответном уведомлении на отправленную команду (см. таблицу 24).
import	Параметры запрашиваемого изображения: • cam — идентификатор камеры, для которой необходимо экспортировать кадр; • time — дата и время кадра. **При экспорте архивного кадра** Значение может быть задано в одном из следующих форматов: – ДД-ММ-ГГ ЧЧ:ММ:СС.XXX; – ГГГГ-ММ-ДД ЧЧ:ММ:СС.XXX. **Примечание.** Миллисекунды указываются опционально (любое количество символов).

	Поиск кадра осуществляется в пределах секунды (справа и слева от указанного времени). Для экспорта берется кадр, ближайший к указанному времени. Например, если указанное время экспорта `10:56:02` (или `10:56:02.xxx`), система будет осуществлять поиск кадра в интервале `10:56:02.000` (`10:56:02.xxx`) - `10:56:02.999`. В случае отсутствия кадра в архиве указанной камеры на данном видеосервере за указанное время, операция завершается с ошибкой. **При экспорте кадра живого видео** Для экспорта кадра живого видео используется значение `live`. Например: `. . . , "import","cam$7;time$live", . . .` В результате выполнения команды будет экспортирован ближайший по времени ключевой кадр (`i`-кадр).
`export_engine`	Тип сохранения. Возможные значения: • `sql` — сохранение в базу данных; • `file` — сохранение в файл.
`export`	Параметры процедуры экспорта. *При сохранении в файл:* • `filename` — имя сохраняемого файла. Расширение имени файла указывается опционально. Если не указано или указано некорректно, расширение подставляется/заменяется системой автоматически, в зависимости от типа исходного файла (см. параметр `export_image`); • `dir` — полный путь к файлу. Допускается задавать как существующую, так и создаваемую директорию. **Примечание.** Результирующий файл должен сохраняться на логических дисках (а так же доступных для записи ресурсах локальной сети) того *Видеосервера*, на котором создан данный объект *Image Processor*. *При сохранении в БД:* • `идентификатор внешней БД` — идентификатор внешней БД, настроенной в SecurOS; • `запрос` — SQL-запрос для сохранения изображения в БД.

`export_image (необязательный)`	Параметры результирующего файла: • `format` — формат сохраняемого файла. Возможные значения: `jpeg`, `jpg`, `png`. Значение по умолчанию — `jpg`; • `quality` — качество изображения. Возможные значения: [`-1; 0 — 100`]. Значение по умолчанию: `-1`, соответствует качеству, заданному в используемой библиотеке экспорта изображений. **Примечание.** Чем выше значение параметра, тем лучше качество изображения и больше его размер.

process (необязательный)	Параметры редактирования изображения (обрезка исходного изображения, нанесение фигур, текста и пр.): • color — цвет объекта, наносимого на изображение. Указывается название или код цвета (в формате RGB, см. https://www.w3.org/TR/css-color-3/#valuea-def-color). Значение по умолчанию: green; • penwidth:N — толщина линии. Возможные значения N (в пикселах): [0;100]. Значение по умолчанию — 2; • rect:x,y,w,h — нарисовать прямоугольник с координатами верхнего левого угла x, y, шириной w и высотой h (в процентах от размера изображения). Возможные значения каждого параметра: [0; 100]; **Примечание.** Если заданные размеры прямоугольника превышают размеры изображения, выходящая за рамки изображения область прямоугольника обрезается. • crop:x,y,w,h — обрезать изображения до размеров w, h (ширина, высота) начиная с левого верхнего угла x, y (в процентах от размеров исходного изображения). Возможные значения каждого параметра: [0; 100]; • default — установить для параметров color и penwidth значения по умолчанию; • font:N — размер шрифта (в пикселах). Возможные значения: [1; 100]. Значение по умолчанию — 11; **Примечание.** В зависимости от используемой библиотеки может задаваться в относительных единицах. • polygon:x1,y1,x2,y2, . . . ,xn,yn — нарисовать многоугольную замкнутую фигуру с координатами вершин x1,y1,x2,y2, . . . ,xn,yn (в процентах от размеров исходного изображения). Вершины соединяются последовательно, прямыми линиями, последняя указанная вершина соединяется с первой; • polyline:x1,y1,x2,y2, . . . ,xn,yn — нарисовать ломаную линию с координатами вершин x1,y1,x2,y2, . . . ,xn,yn (в процентах от размеров исходного изображения). Вершины соединяются последовательно, прямыми линиями; • text:x,y,text — нанести текст на изображение. Начало ввода текста в точке с координатами x, y (нижняя левая граница первого символа, в пикселах); • reltext:x,y,text — нанести текст на изображение. Начало ввода текста в точке с координатами x, y (нижняя левая граница первого символа, в % от размера изображения);

	• `scale:w,h` — масштабировать исходное изображение до размеров `w, h` (ширина, высота, в пикселах). При этом выполняются следующие правила: – Если указан только один из аргументов, масштабирование выполняется до указанного размера с сохранением исходных пропорций; – Если указаны оба аргумента, масштабирование выполняется до указанных размеров без сохранения исходных пропорций. **Примечания:** 1. В случае, если вводимая строка содержит символ `";"` (точка с запятой), весь текст должен быть экранирован символами `"$$. . . $$"`, например, `"$$один;два$$"`. 2. В случае, если длина строки или размер шрифта превышает размер кадра, на изображение наносится часть надписи, соответствующая размеру кадра. 3. Список шрифтов и языков зависит от установленной операционной системы. **Внимание!** В одной команде экспорта допускается указывать произвольное число параметров редактирования изображения. Указанные параметры обрабатываются последовательно, т.е. каждый следующий параметр применяется к изображению, полученному в результате выполнения команды с предыдущим указанным параметром.
`caption`	• `text_size:text` — добавить под результирующий кадр, полученный после выполнения операций `process`, область черного цвета и вывести в ней `text` белого цвета и размера шрифта `text_size` (необязательный параметр, если не указан, используется значение по умолчанию). Ширина добавляемой области устанавливается равной ширине результирующего кадра. Высота области устанавливается автоматически и зависит от длины строки исходного текста. **Примечания:** 1. В случае, если исходная строка имеет длину большую, чем ширина изображения, система автоматически переносит текст на следующую строку "по словам". В случае, если единственное слово не помещается на строке, система обрезает символы слова по ширине изображения. 2. В исходной строке `text` допускается использование символов табуляции и перевода строки. Применимо к Модулям и сторонним интеграциям.

Идентификатор действия: `CLEAR_QUEUE`

Название в макрокоманде: нет

Описание: Удаление всех заданий для всех *Камер* из очереди заданий

Параметры: нет

Листинг 13. Пример команды `EXPORT`

```
const sos=require('securos');
sos.connect(async function(core)
{
      core.registerEventHandler("MACRO", "1.3", "RUN", () =>
       //какая макрокоманда запускает
      {
            let params = {
                  "import":"cam$1;time$2023-05-03 10:45:39.000",
                  "export_engine":"file",
                  "export":"filename$ex_frame;
                  dir$C:\\EXPORTED_FRAMES_EXAMPLE",
                  "export_image":"format$png;quality$70",
                  "process":"rect:10,10,30,40;color:255,0,0;rect:5,5,20,20;
                  color:blue;rect15,15,20,20;default;penwidth:5;
                  rect25,25,20.5,20;crop:0,0,50,55.5"
                  };
            core.doReact("IMAGE_EXPORT", "1", "EXPORT", params);
      });
});

/* В результате выполнения команды создается файл
C:\EXPORTED_FRAMES_EXAMPLE\ex_frame.png
На изображение наносятся прямоугольники (координаты даны в %
от исходного изображения):
зеленый 10,10,30,40, красный 5,5,20,20, синий 15,15,20,20
с толщиной линии 3,
зеленый 25,25,20.5,20 с толщиной линии 5.
Затем изображение обрезается */
```

4.3.8 Контроллер Видеостены

Идентификатор типа объекта: `VW_CONTROLLER`.

Таблица 25. События `VW_CONTROLLER`

Идентификатор события: `LAYOUT_ACTIVATED`	
Название в макрокоманде: Текущая раскладка была изменена	
Описание: Текущая раскладка видеостены была изменена	
Параметры:	
`id`	Название активированной раскладки
`user_id`	Идентификатор пользователя SecurOS или идентификатор пользователя Windows при использовании **Active Directory** (см. **Руководство администратора SecurOS**), сменившего текущую раскладку
`user_name`	Имя пользователя SecurOS или имя пользователя Windows при использовании **Active Directory** (см. **Руководство администратора SecurOS**), сменившего текущую раскладку

4.4 Аудиоподсистема

В данном разделе описаны события и действия объектов аудиоподсистемы.

4.4.1 Микрофон

Идентификатор типа объекта: `AUDIO_LINE`.

Таблица 26. События `AUDIO_LINE`

Идентификатор события: `ARM`
Название в макрокоманде: Запись включена
Описание: Включен датчик звука
Параметры: нет
Идентификатор события: `DISARM`
Название в макрокоманде: Запись выключена
Описание: Выключен датчик звука
Параметры: нет
Идентификатор события: `REC`
Название в макрокоманде: Начало записи
Описание: Начало записи
Параметры: нет
Идентификатор события: `REC_STOP`
Название в макрокоманде: Конец записи
Описание: Конец записи
Параметры: нет
Идентификатор события: `RESET`
Название в макрокоманде: Подключение микрофона
Описание: Подключение микрофона
Параметры: нет
Идентификатор события: `VOX_OFF`
Название в макрокоманде: Выключение акустопуска
Описание: Выключение акустопуска

Параметры: нет
Идентификатор события: VOX_ON
Название в макрокоманде: Включение акустопуска
Описание: Включение акустопуска
Параметры: нет

Таблица 27. Действия AUDIO_LINE

Идентификатор действия: ARM
Название в макрокоманде: Поставить на охрану
Описание: Начать запись **Примечание.** Если в настройках объекта *Микрофон* задан параметр **Порог** (см. **Руководство администратора SecurOS**), запись начнется только при достижении заданного уровня звука. В противном случае запись начнется сразу после выполнения команды.
Параметры: нет
Идентификатор действия: DISARM
Название в макрокоманде: Снять с охраны
Описание: Остановить запись
Параметры: нет

4.5 Подсистема ввода/вывода

В данном разделе описаны события и действия объектов подсистемы ввода/вывода.

4.5.1 Луч

Идентификатор типа объекта: GRAY.

Таблица 28. События GRAY

Идентификатор события: ALARM
Название в макрокоманде: Тревога
Описание: Зафиксирована тревога
Параметры:

`comment`	Содержит идентификаторы *Камер* (`cams`), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.
Идентификатор события: `ARM`	
Название в макрокоманде: Поставлен на охрану	
Описание: Поставлен на охрану	
Параметры:	
`comment`	Содержит идентификаторы *Камер* (`cams`), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.
Идентификатор события: `CONFIRM`	
Название в макрокоманде: Тревога принята	
Описание: Тревога подтверждена	
Параметры:	
`comment`	Содержит идентификаторы *Камер* (`cams`), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.
Идентификатор события: `DISARM`	
Название в макрокоманде: Снят с охраны	
Описание: Снят с охраны	
Параметры:	
`comment`	Содержит идентификаторы *Камер* (`cams`), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.
Идентификатор события: `NOT_VALID_STATE`	
Название в макрокоманде: Зона не готова	
Описание: Зона не готова	
Параметры:	
`comment`	Содержит идентификаторы *Камер* (`cams`), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.
Идентификатор события: `OPEN`	
Название в макрокоманде: Обрыв	
Описание: Обрыв	

Параметры:	
comment	Содержит идентификаторы *Камер* (cams), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.
Идентификатор события: NORM	
Название в макрокоманде: Зона готова	
Описание: Зона готова	
Параметры:	
comment	Содержит идентификаторы *Камер* (cams), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.
Идентификатор события: SHORT_CIRCUIT	
Название в макрокоманде: Короткое замыкание	
Описание: Короткое замыкание	
Параметры:	
comment	Содержит идентификаторы *Камер* (cams), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.
Идентификатор события: SABOTAGE	
Название в макрокоманде: Саботаж	
Описание: Саботаж	
Параметры:	
comment	Содержит идентификаторы *Камер* (cams), выбранных в настройках объекта *Луч*. Подробнее см. раздел **Использование параметра comment для визуализации событий**.

Таблица 29. Действия GRAY

Идентификатор действия: DISABLE_RAY	
Название в макрокоманде: Отключить датчик	
Описание: Отключить объект, соответствующий датчику	
Параметры:	
parent_id	Идентификатор родительского устройства видеозахвата
Идентификатор действия: ARM	

Название в макрокоманде: Поставить на охрану
Описание: Поставить на охрану
Параметры: нет
Идентификатор действия: `CONFIRM`
Название в макрокоманде: Принять тревогу
Описание: Подтвердить тревогу
Параметры: нет
Идентификатор действия: `DISARM`
Название в макрокоманде: Снять с охраны
Описание: Снять с охраны
Параметры: нет

4.5.2 Реле

Идентификатор типа объекта: `GRELE`.

Таблица 30. События `GRELE`

Идентификатор события: `OFF`
Название в макрокоманде: Реле выключено
Описание: Реле выключено
Параметры: нет
Идентификатор события: `ON`
Название в макрокоманде: Реле включено
Описание: Реле включено
Параметры: нет

Таблица 31. Действия `GRELE`

Идентификатор действия: `OFF`
Название в макрокоманде: Выключить
Описание: Выключить реле
Параметры: нет

Идентификатор действия: ON	
Название в макрокоманде: Включить	
Описание: Включить реле	
Параметры:	
timeout	
Идентификатор действия: DISABLE_RELE	
Название в макрокоманде: Отключить реле	
Описание: Отключить объект, соответствующий реле	
Параметры:	
parent_id	Идентификатор родительского устройства видеозахвата

4.6 Подсистема оповещения

В данном разделе описаны события и действия следующих объектов подсистемы оповещения:

- **Сервис почтовых сообщений**.
- **Почтовое сообщение**.
- **Короткое сообщение**.
- **Сервис звукового оповещения**.
- **Служба реагирования**.
- **SIP-устройство**.

4.6.1 Сервис почтовых сообщений

Идентификатор типа объекта: MMS.

Таблица 32. События MMS

Идентификатор события: SENT	
Название в макрокоманде: нет	
Описание: Сообщение отправлено	
Параметры:	
request_id	Идентификатор исходного сообщения
Идентификатор события: SEND_ERROR	
Название в макрокоманде: нет	
Описание: Ошибка отправки сообщения	

Параметры:	
request_id	Идентификатор исходного сообщения

Таблица 33. Действия MMS

Идентификатор действия: SEND	
Название в макрокоманде: нет	
Описание: Запрос на отправку сообщения с кодом доступа для двухфакторной аутентификации	
Параметры:	
subject	Тема сообщения
body	Тело сообщения
from	Адрес электронной почты отправителя
to	Адрес электронной почты получателя
cc	Адрес электронной почты получателей копии письма. Необязательный параметр
request_id	Произвольный идентификатор сообщения. Указание идентификатора позволяет соотнести ответное письмо об успехе или ошибке с исходным письмом. Необязательный параметр
use_html (опциональный)	Параметр определяет формат, в котором будет отправлено сообщение. Возможные значения: • 0 — сообщение будет отправлено без форматирования. Значение по умолчанию. • 1 — сообщение будет отправлено в формате HTML.

4.6.2 Почтовое сообщение

Идентификатор типа объекта: MAIL_MESSAGE.

Таблица 34. События MAIL_MESSAGE

Идентификатор события: SEND_ERROR (alarm)
Название в макрокоманде: Ошибка отправки сообщения
Описание: Ошибка отправки сообщения
Параметры: нет
Идентификатор события: SENT

Название в макрокоманде: Сообщение отправлено
Описание: Сообщение отправлено
Параметры: нет

Таблица 35. Действия `MAIL_MESSAGE`

Идентификатор действия: `SEND`	
Название в макрокоманде: Отправить сообщение	
Описание: Отправка сообщения по заданному шаблону	
Параметры:	
`param_name_1, param_name_2, ..., param_name_N`	Произвольные переменные параметры сообщения, заданные в *Теме* и/или *Теле* сообщения. Необязательные параметры
`attachments`	Переопределяемый в команде параметр, заданный в поле **Приложение** шаблона почтового сообщения. Необязательный параметр

4.6.3 Короткое сообщение

Идентификатор типа объекта: `SHORT_MESSAGE`.

Таблица 36. События `SHORT_MESSAGE`

Идентификатор события: `SEND_ERROR` (alarm)
Название в макрокоманде: Ошибка отправки сообщения
Описание: Ошибка отправки сообщения
Параметры: нет
Идентификатор события: `SENT`
Название в макрокоманде: Сообщение отправлено
Описание: Сообщение отправлено
Параметры: нет

Таблица 37. Действия `SHORT_MESSAGE`

Идентификатор действия: `SEND`
Название в макрокоманде: Отправить сообщение
Описание: Отправить сообщение

Параметры:	
`text`	Произвольный текст, отличный от заданного в шаблоне сообщения (параметр **Сообщение** в настройках объекта *Короткое сообщение*). Максимальная длина сообщения — `70` символов (латиница/кириллица). Возможные значения: • `<user_text>` — будет отправлено сообщение с пользовательским текстом; • `<NULL (значение_не_задано)>` — будет отправлено пустое сообщение; **Внимание!** Если параметр не указан, будет отправлено сообщение с шаблонным текстом.

4.6.4 Сервис звукового оповещения

Идентификатор типа объекта: `VNS`.

События: отсутствуют.

Таблица 38. Действия `VNS`

Идентификатор действия: `PLAY_WAV`	
Название в макрокоманде: Проиграть звуковой файл	
Описание: Воспроизведение звука	
Параметры:	
`file`	Путь к `WAV`-файлу относительно корневого каталога SecurOS, включающий в себя расширение файла (например, `"file","\\wav\\cam_alarm_1.wav"`)

4.6.5 Служба реагирования

Идентификатор типа объекта: `EMERGENCY`.

Таблица 39. События `EMERGENCY`

Идентификатор события: `SUCCESS`
Название в макрокоманде: Карточка отправлена успешно
Описание: Карточка отправлена успешно
Параметры:

comment	Структура, включающая следующую информацию: • идентификатор *Карточки происшествия*; • ответ http-сервера (внутренний идентификатор *Карточки происшествия*).
Идентификатор события: FAILED	
Название в макрокоманде: Не удалось отправить Карточку	
Описание: Не удалось отправить Карточку	
Параметры:	
comment	Структура, включающая следующую информацию: • идентификатор *Карточки происшествия*; • код отклика и ответ http-сервера (или прочее описание причины ошибки); • данные о происшествии (в формате JSON).

Команды: отсутствуют

4.6.6 SIP-устройство

Идентификатор типа объекта: SIP_DEVICE.

Таблица 40. События SIP_DEVICE

Идентификатор события: INCOMING_CALL	
Название в макрокоманде: Входящий звонок	
Описание: Поступил входящий звонок от этого SIP-устройства	
Параметры: нет	
Идентификатор события: OUTGOING_CALL	
Название в макрокоманде: Исходящий звонок	
Описание: Оператор совершил исходящий звонок на это SIP-устройство	
Параметры:	
comment	Структура, включающая следующую информацию: • имя оператора, совершившего звонок; • имя компьютера, за которым находился оператор.

Команды: отсутствуют

4.7 Подсистема автоматизации

В данном разделе описаны события и действия объектов подсистемы автоматизации.

4.7.1 Расписание

Идентификатор типа объекта: `TIME_ZONE`.

Таблица 41. События `TIME_ZONE`

Идентификатор события: `ACTIVATE`
Название в макрокоманде: Начало
Описание: Начало действия расписания
Параметры: нет
Идентификатор события: `DEACTIVATE`
Название в макрокоманде: Конец
Описание: Окончание действия расписания
Параметры: нет

Действия: отсутствуют.

4.7.2 Макрокоманда

Идентификатор типа объекта: `MACRO`.

Таблица 42. События `MACRO`

Идентификатор события: `RUN`
Название в макрокоманде: Выполнено действие
Описание: Макрокоманда выполнена
Параметры: нет

Таблица 43. Действия MACRO

Идентификатор действия: RUN
Название в макрокоманде: Выполнить действие
Описание: Выполнить макрокоманду
Параметры: нет

4.8 Модули специального назначения

В данном разделе описаны события и действия модулей специального назначения.

4.8.1 Термометр

Идентификатор типа объекта: THERMOMETER.

Таблица 44. События THERMOMETER

Идентификатор события: MEASURE_RESULT	
Название в макрокоманде: Нет	
Описание: Отправляется каждый раз при успешном совмещении рамки лица, распознанного модулем SecurOS FaceX, с показаниями тепловизора	
Параметры:	
thermal_array	Массив измеренных температур. Массив состоит из значений, встречаемых в области температур, и количества точек с такой температурой. Например: (36.6, 20), (36.3, 4), (36.9, 2).
Идентификатор события: TEMP_EXCEEDED	
Название в макрокоманде: Нет	
Описание: Событие отправляется, если в массиве измеренных температур есть хотя бы одно значение, превышающее пороговое значение температуры, заданное параметром **Максимально допустимая температура** в настройках объекта *Термометр* (см. **Руководство администратора SecurOS**).	
Параметры:	
facex_server_id	Идентификатор используемого *FaceX: Сервера*
channel_id	Идентификатор *Камеры* FaceX
best_frame_timestamp	Дата и время кадра, на котором лицо, обнаруженное модулем SecurOS FaceX, видно наилучшим образом
visualization	Набор параметров в формате команды модуля **Image Processor**. Используется для отрисовки рамки вокруг объекта детектирования при просмотре кадра с помощью *Медиа Клиента*

`comment`	Комментарий к событию
`mask`	Наличие маски на распознанной персоне. Может принимать следующие значения: • `0` — маска отсутствует; • `1` — маска присутствует.
`track_id`	Идентификатор трека текущей персоны
`detection_id`	Идентификатор процедуры распознавания лица персоны модулем SecurOS FaceX

Команды: отсутствуют

5 Приложение 2. События и действия сервисов SecurOS

В данном разделе описаны события и действия сервисов SecurOS.

5.1 EDGE_STORAGE

Идентификатор типа сервиса: `EDGE_STORAGE`.

События: нет

Таблица 45. Действия `EDGE_STORAGE`

Идентификатор действия: `SYNC_NOW`
Название в макрокоманде: нет
Описание: Запуск процедуры синхронизации архивов
Параметры: нет

6 Приложение 3. Веб-серверы модулей SecurOS

В SecurOS есть модули, имеющие в своем составе собственный веб-сервер:

- REST API (см. **Руководство программиста REST API**).
- SecurOS Auto (см. **Руководство программиста REST API**).

Примечание. **Руководство программиста REST API** не входит в общий комплект документации и предоставляется по запросу.

- **Health Monitor**.

6.1 Health Monitor

Для получения сообщений о проблемах *Видеосервера* с помощью API-методов, на каждом из *Видеосерверов* реализован собственный WebSocket-сервер. Такой сервер автоматически разворачивается при запуске *Видеосервера*.

Для получения информации о проблемах *Видеосервера* необходимо:

1. **Подключиться к Видеосерверу по протоколу WebSocket**.
2. **Пройти процедуру авторизации**.

Подключение к Видеосерверу

Чтобы подключиться к *Видеосерверу* по протоколу WebSocket используйте URL следующего вида:

`wss://<IP-адрес_Видеосервера>:8092`, где:

- `wss` — протокол подключения к *Видеосерверу*;
- `<IP-адрес_Видеосервера>` — IP-адрес *Видеосервера*;
- `8092` — номер порта *Видеосервера* для подключения. В SecurOS данное значение является фиксированной величиной.

Примечание. Если необходимо получать информацию от каждого *Видеосервера* сети SecurOS, последовательно подключитесь к каждому из них.

При успешном подключении устанавливается соединение с WebSocket-сервером *Видеосервера*.

Авторизация

После того, как соединение установлено, необходимо авторизоваться на *Видеосервере*. Для авторизации необходимо отправить на *Видеосервер* следующий запрос:

Пример запроса

```
{
  "method": "subscribe",
  "user": "root",
  "password": "securos"
}
```

Параметры запроса

- **method** — тип запроса `subscribe` (подписка на события). Обязательный параметр;
- **user** — имя пользователя. Обязательный параметр;
- **password** — пароль пользователя. Обязательный параметр.

Примечание. Для авторизации можно использовать данные учетной записи суперпользователя SecurOS.

Возможные ответы:

- `{"success":1}` — успешная авторизация;
- `{"success":0, "error":"Invalid message"}` — получен пустой или некорректный JSON;
- `{"success":0, "error":"Invalid method"}` — неправильно указано значение параметра **method** (может быть только `subscribe`);
- `{"success":0, "error":"Empty user"}` — пустое имя пользователя или параметр пропущен;
- `{"success":0, "error":"Wrong user or password"}` — неудачная авторизация.

При успешной авторизации клиент сразу получает список всех текущих проблем данного сервера. После этого будут приходить сообщения о новых проблемах этого сервера или закрытии старых проблем этого сервера. В дальнейшем, чтобы получить список текущих проблем, нужно повторно отправить запрос на авторизацию.

Пример сообщения о проблеме

```
{
    "action": "open",
    "description": "Сообщение появляется, если не удается получить видеопоток, выбранный
в параметре \"Поток для записи\" в настройках <i>Камеры</i>. Когда недоступен \"Поток
для записи\", все остальные потоки отключаются.<br>Сообщение пропадает, если поток с
корректным содержимым удается получать в течение 7 секунд.",
    "problem": "Camera is detached",
    "source": "cameras/2",
    "time": "2021-11-10T12:59:28"
}
```

Параметры сообщения о проблеме

- **action** — стадия проблемы. Может принимать следующие значения:
  - `open` — проблема появилась;
  - `close` — проблема исчезла.
- **description** — описание проблемы;
- **problem** — название проблемы;
- **source** — название группы проблемных объектов и идентификатор проблемного объекта;

- **time** — дата и время возникновения проблемы. Данное значение указывается в сообщении как при возникновении проблемы, так и при ее закрытии.

7 Информация для Службы технической поддержки

Данный раздел описывает требования к служебной информации, необходимой при обращении в Службу технической поддержки компании Intelligent Security Systems.

Примечание. Собранные сведения необходимо направлять в Службу технической поддержки (см. раздел **Обращение за технической поддержкой**).

Для более скорого разрешения проблем подготовьте следующую техническую информацию:

Внимание! Сведения в пунктах, отмеченных знаком "*", являются обязательными для предоставления.

1. (*) Ф. И. О.
2. (*) Название организации.
3. (*) Контактная информация: телефон, e-mail.
4. Если Вы являетесь партнером Intelligent Security Systems, то укажите, с каким менеджером компании Intelligent Security Systems Вы работаете; в ином случае, укажите следующие сведения:
   - Компания, в которой приобретался комплект программного и аппаратного обеспечения.
   - Действия для устранения проблемы, предложенные при обращении к партнеру, у которого приобретался комплект.
5. (*) Описание проблемы (неполадки).
6. (*) Описание действий, которые приводят к возникновению проблемы.
7. Описание изменений в настройках/конфигурации системы, которые привели к возникновению проблемы.
8. Системная и диагностическая информация о компьютере и конфигурации системы SecurOS, полученная с помощью утилиты **SystemInfo** (см. **Руководство администратора SecurOS** для подробной информации об использовании утилиты).

   Если невозможно запустить данную утилиту, предоставьте следующую информацию:

   - (*) идентификаторы и даллас-коды используемых ключей Guardant;

Примечание. Даллас-коды оборудования можно просмотреть с помощью утилиты **Hardware Report Utility** (см. **Руководство администратора SecurOS** для подробной информации об использовании утилиты).

   - (*) наименование и версия установленного ПО производства компании Intelligent Security Systems;
   - общее количество видеосерверов и удаленных рабочих мест оператора в системе;
   - операционная система (наименование платформы, версия сервисного пакета).

9. По возможности предоставьте любую другую полезную информацию, например:
   - конфигурация компьютерного оборудования;
   - загрузка центральных процессоров;
   - объемы используемой оперативной и виртуальной памяти;
   - загрузка сети;
   - конфигурация сети и сетевого окружения.

Предметный указатель

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

А

В

З

К

Л

М

О

П

Р

С

Т