ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
СИСТЕМЫ КОНТРОЛЯ И УПРАВЛЕНИЯ ДОСТУПОМ
«U-Prox IP»

Web-сервис U-Prox IP

Описание операций

Версия 3.035

Производство: ООО «А2СОФТИН ЛТД», Украина, 03035,
Киев, ул. Митрополита Василия Липковского, д.1
Тел: +380(44)248-6588, +380(44)248-6589

В связи с непрерывным развитием системы, ее аппаратного и программного обеспечения, данный документ подвержен периодическим изменениям и дополнениям. Самую последнюю версию Вы можете загрузить с WEB-сайта продукта по адресу http://u-prox.com/.

Права и их защита

1. Всеми правами на данный документ и связанные с ним иллюстрации обладает компания «А2СОФТИН». Не допускается копирование, перепечатка и любой другой способ воспроизведения документа или его части без согласия «А2СОФТИН».

2. Представленные в документе иллюстрации и данные являются типичными примерами и должны быть специально подтверждены «А2СОФТИН» перед оформлением любых тендеров, заказов и контрактов.

Сертификация

Содержание

Общие сведения

Web-сервис U-Prox IP предназначен для организации взаимодействия ПО СКУД «U-Prox IP» с web-приложениями или другими информационными системами.

Web-сервис U-Prox IP регистрируется в системе во время установки и обеспечивает доступ к базе данных СКУД «U-Prox IP» через программные интерфейсы REST API и SOAP API, а также через пользовательский web-интерфейс.

Состав

Во время развертывания сервиса на сервере устанавливаются следующие файлы и каталоги:

Acs.WebService.exe	WEB-сервис
*.dll	Библиотеки, необходимые для работы web-сервиса
Web Pages*.*	Файлы пользовательского web-интерфейса

В список исключений брандмауэра Windows добавляется порт 40001, через который осуществляется подключение к web-сервису.

Установка web-сервиса

Установка web-сервиса осуществляется по команде пользователя с помощью программы установки ПО СКУД «U-Prox IP».

Удаление web-сервиса

Удаление web-сервиса выполняется средствами операционной системы после выбора web-сервиса U-Prox IP в списке установленных программ в окне параметров Windows (из Панели Управления WIndows).

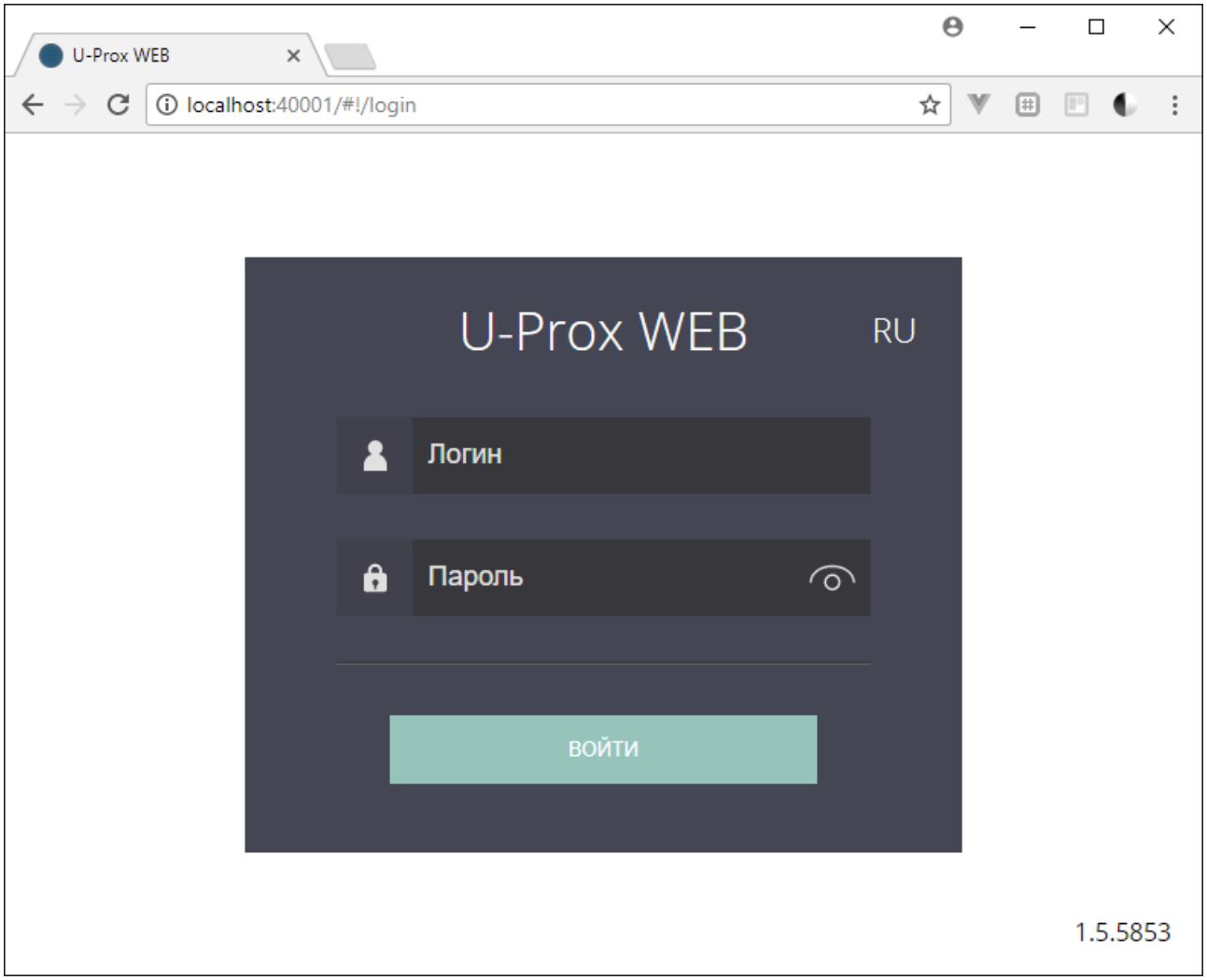
Последовательность работы с web-сервисом

Запуск web-сервиса

Web-сервис загружается автоматически в режиме отложенного старта – в течение двух минут после загрузки операционной системы Windows.

Первый вход в систему

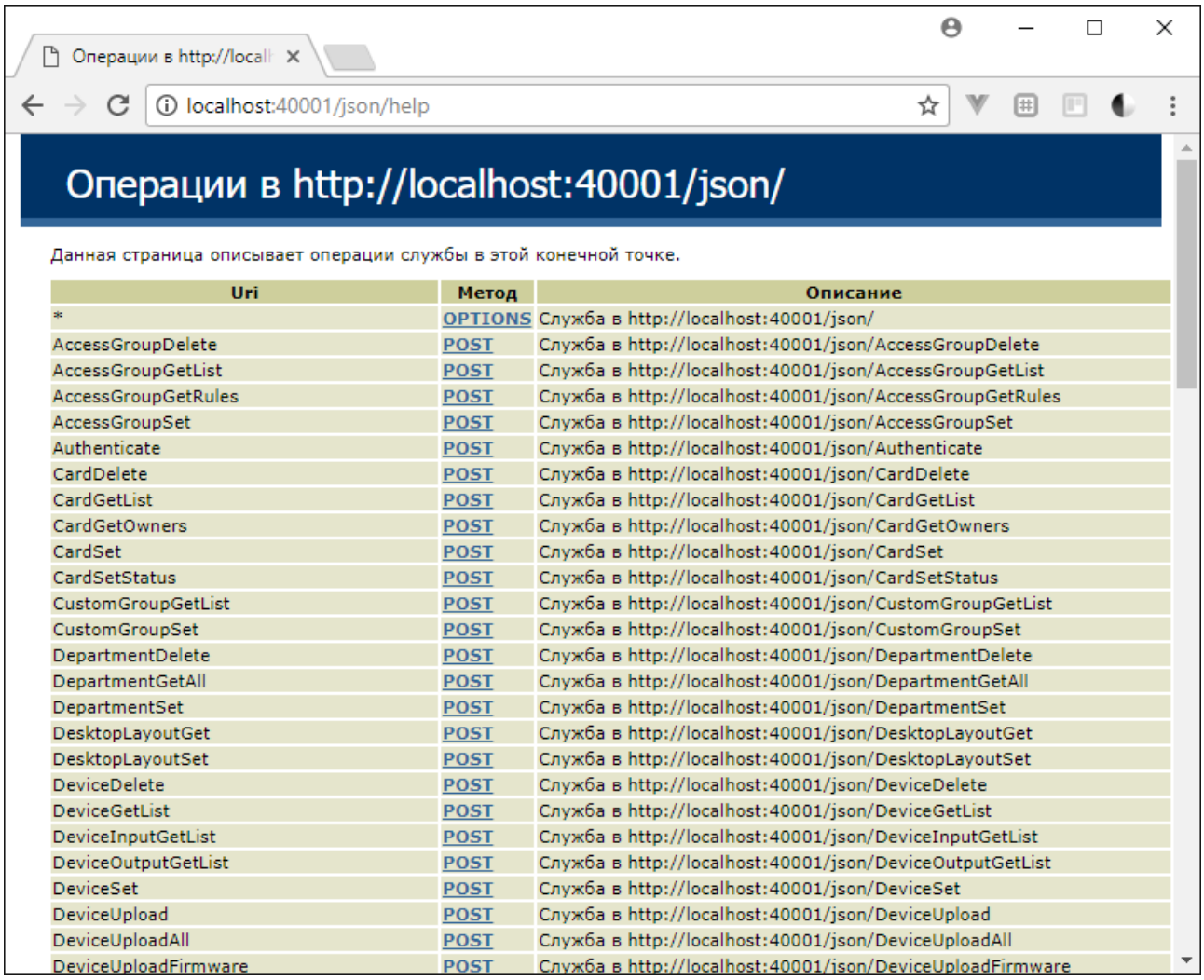
После установки web-сервиса на рабочем столе пользователя создается ярлык «U-Prox IP Web». При выборе этого ярлыка открывается окно браузера со страницей по адресу http://localhost:40001/:



Войти в систему можно, указав логин **admin** и пароль **admin**. Настройка логинов, паролей и прав доступа к web-сервису осуществляется на странице роли Администратора. Настройка оборудования осуществляется на странице роли Инсталлятора. Другие роли предназначены для просмотра и фото-верификации событий СКУД, настройки расписаний и прав доступа пользователей СКУД, а также составления отчетов.

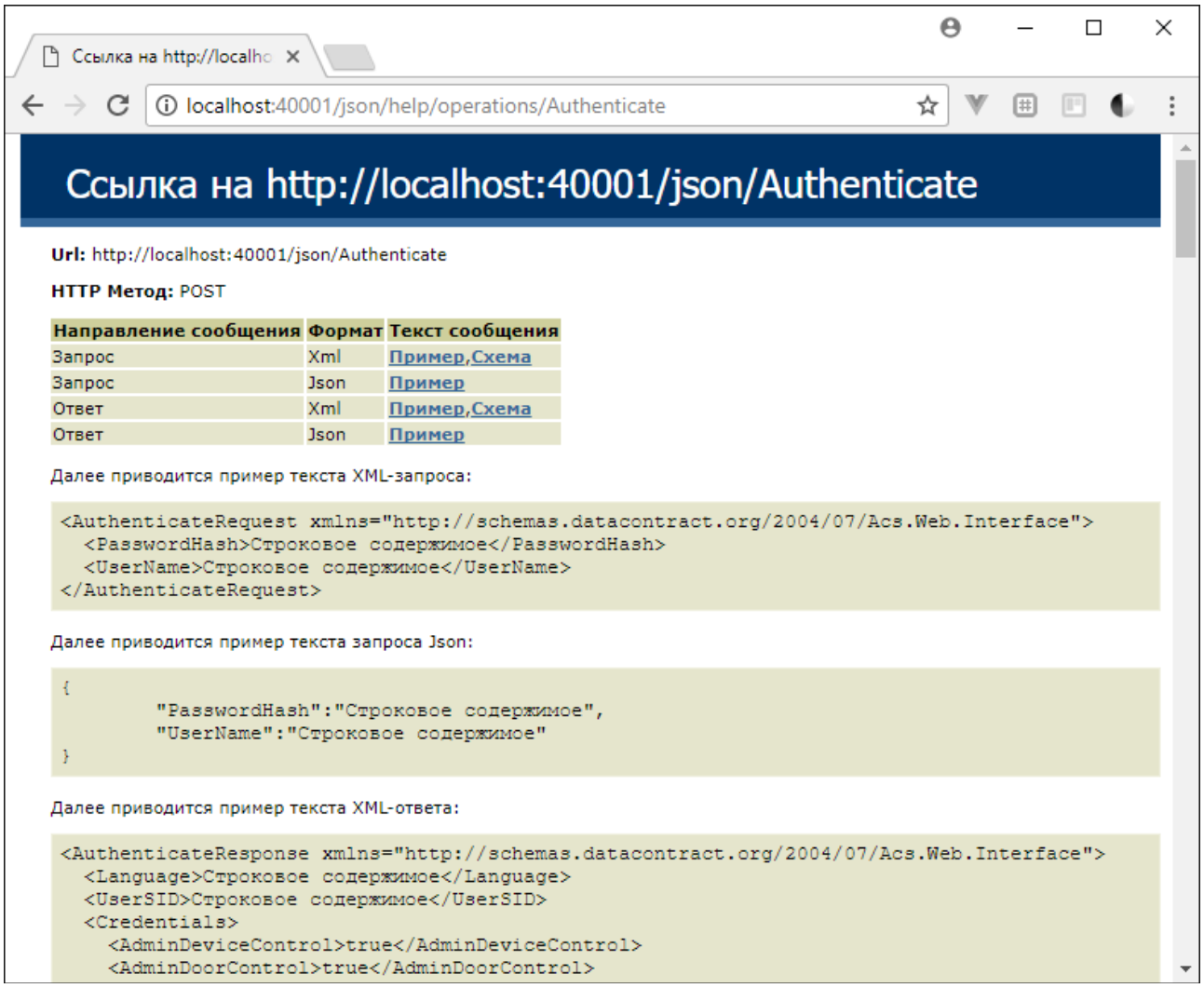
Краткий перечень операций REST API и SOAP API

Перечень поддерживаемых операций REST API и SOAP API отображается на странице по адресу http://localhost:40001/json/help:



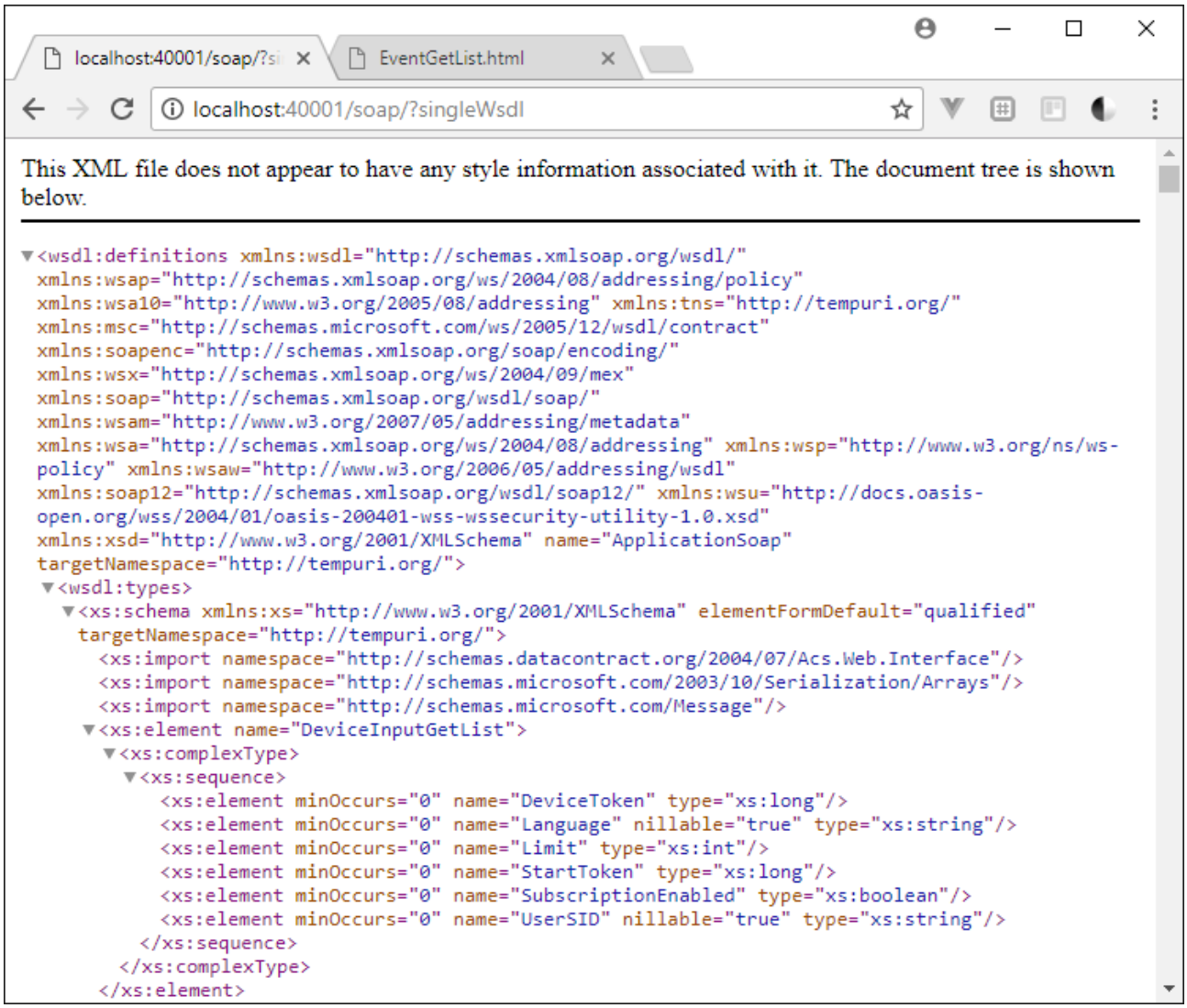
Uri	Метод	Описание
*	OPTIONS	Служба в http://localhost:40001/json/
AccessGroupDelete	POST	Служба в http://localhost:40001/json/AccessGroupDelete
AccessGroupGetList	POST	Служба в http://localhost:40001/json/AccessGroupGetList
AccessGroupGetRules	POST	Служба в http://localhost:40001/json/AccessGroupGetRules
AccessGroupSet	POST	Служба в http://localhost:40001/json/AccessGroupSet
Authenticate	POST	Служба в http://localhost:40001/json/Authenticate
CardDelete	POST	Служба в http://localhost:40001/json/CardDelete
CardGetList	POST	Служба в http://localhost:40001/json/CardGetList
CardGetOwners	POST	Служба в http://localhost:40001/json/CardGetOwners
CardSet	POST	Служба в http://localhost:40001/json/CardSet
CardSetStatus	POST	Служба в http://localhost:40001/json/CardSetStatus
CustomGroupGetList	POST	Служба в http://localhost:40001/json/CustomGroupGetList
CustomGroupSet	POST	Служба в http://localhost:40001/json/CustomGroupSet
DepartmentDelete	POST	Служба в http://localhost:40001/json/DepartmentDelete
DepartmentGetAll	POST	Служба в http://localhost:40001/json/DepartmentGetAll
DepartmentSet	POST	Служба в http://localhost:40001/json/DepartmentSet
DesktopLayoutGet	POST	Служба в http://localhost:40001/json/DesktopLayoutGet
DesktopLayoutSet	POST	Служба в http://localhost:40001/json/DesktopLayoutSet
DeviceDelete	POST	Служба в http://localhost:40001/json/DeviceDelete
DeviceGetList	POST	Служба в http://localhost:40001/json/DeviceGetList
DeviceInputGetList	POST	Служба в http://localhost:40001/json/DeviceInputGetList
DeviceOutputGetList	POST	Служба в http://localhost:40001/json/DeviceOutputGetList
DeviceSet	POST	Служба в http://localhost:40001/json/DeviceSet
DeviceUpload	POST	Служба в http://localhost:40001/json/DeviceUpload
DeviceUploadAll	POST	Служба в http://localhost:40001/json/DeviceUploadAll
DeviceUploadFirmware	POST	Служба в http://localhost:40001/json/DeviceUploadFirmware

В случае перехода на страницу метода POST какой-либо операции отображаются примеры текстов запросов и ответов этой операции:



Документ WSDL для SOAP API

Документ WSDL 1.1 для SOAP API доступен по ссылке http://localhost:40001/soap/?singleWsdl:



Описание операций

Authenticate

Назначение

Начало сеанса работы с web-сервисом

Url: http://localhost:40001/json/Authenticate

HTTP Метод: POST

Текст запроса Json:

```
{
	"PasswordHash":"Строковое содержимое",
	"UserName":"Строковое содержимое"
}
```

Свойства запроса

UserName
	Логин пользователя
PasswordHash
	значение хэш-функции пароля пользователя:

	PasswordHash = MD5(MD5(MD5(пароль пользователя) + “F593B01C562548C6B7A31B30884BDE53”))

	Функция MD5(…) должна возвращать значение стандартной хэш-функции MD5 в виде шестнадцатеричной записи с использованием десятичных цифр и символов верхнего регистра: 0..9, A..F.

Текст ответа Json:

```
{
	"UserSID":"Строковое содержимое",
	"Credentials":{
		"AdminDeviceControl":true,
		"AdminDoorControl":true,
		"AdminOperatorChanging":true,
		"EmployeesReports":true,
		"EventsReports":true,
		"HardwareReports":true,
		"InstallatorDataUploading":true,
		"InstallatorDeviceChanging":true,
		"InstallatorDoorChanging":true,
		"InstallatorVersionControl":true,
```

```
                "PassOfficeDataUploading":true,
                "PassOfficeEmployeeControl":true,
                "PassOfficeIdentifierEnrollment":true,
                "PassOfficeVisitorControl":true,
                "PersonnelAccess":true,
                "PersonnelDataUploading":true,
                "PersonnelDepartments":true,
                "PersonnelEmployeeSetting":true,
                "PersonnelSchedules":true,
                "PersonnelUnusedIdentifiers":true,
                "PersonnelVisitorSetting":true,
                "SecurityDoorControl":true,
                "SecurityPhotoverification":true,
                "SecurityViewEvents":true,
                "SystemStatusReports":true,
                "TimeAttendanceReports":true,
                "VisitorsReports":true
        }
}
```


Свойства ответа

UserSID

Уникальный код, используемый для идентификации сеанса пользователя при выполнении последующих запросов

Credentials

Признаки наличия у пользователя прав на выполнение различных груп операций web-сервиса

AdminDeviceControl	Администратор: Загрузка конфигурации в устройства
AdminDoorControl	Администратор: Управления дверьми, турникетами и лифтами
AdminOperatorChanging	Администратор: Изменение операторов и их прав доступа
EmployeesReports	Отчеты: Составление отчетов по персоналу
EventsReports	Отчеты: Составление отчетов по событиям
HardwareReports	Отчеты: Составление отчетов по оборудованию
InstallatorDataUploading	Инсталлятор: Загрузка конфигурации в устройства
InstallatorDeviceChanging	Инсталлятор: Изменение параметров устройств
InstallatorDoorChanging	Инсталлятор: Изменение параметров дверей, турникетов и лифтов
InstallatorVersionControl	Инсталлятор: Обновление микропрограммы устройств
PassOfficeDataUploading	Бюро пропусков: Загрузка конфигурации в устройства
PassOfficeEmployeeControl	Бюро пропусков: Редактирование сотрудников
PassOfficeIdentifierEnrollment	Бюро пропусков: Регистрация идентификаторов
PassOfficeVisitorControl	Бюро пропусков: Редактирование посетителей

PersonnelAccess	Управление персоналом: Редактирование прав доступа
PersonnelDataUploading	Управление персоналом: Загрузка конфигурации в контроллеры
PersonnelDepartments	Управление персоналом: Изменение структуры отделов организации
PersonnelEmployeeSetting	Управление персоналом: Редактирование сотрудников
PersonnelSchedules	Управление персоналом: Редактирование расписаний
PersonnelUnusedIdentifiers	Управление персоналом: Редактирование неиспользуемых идентификаторов
PersonnelVisitorSetting	Управление персоналом: Редактирование посетителей
SecurityDoorControl	Охранник: управление дверьми
SecurityPhotoverification	Охранник: фото-верификация
SecurityViewEvents	Охранник: мониторинг событий
SystemStatusReports	Отчеты: Составление отчетов по состоянию системы
TimeAttendanceReports	Отчеты: Составление отчетов по рабочему времени
VisitorsReports	Отчеты: Составление отчетов по посетитялям

Описание

Для получения доступа к операциям web-сервиса приложение должно аутентифицировать пользователя, от имени которого оно запущена. После установки web-сервиса создается пользователь **admin** с паролем **admin**, которому соответствует значение хэш-функции **88020F057FE7287D8D57494382356F97**.

После успешной аутентификации пользователя web-сервис возвращает *UserSID* – уникальный код сеанса пользователя, который необходимо передавать в качестве параметра при выполнении других операций web-сервиса. Возвращается также перечень признаков наличия у пользователя прав на выполнение различных операций web-сервиса.

В случае неуспешной аутентификации пользователя возвращается *UserSID* с пустым значением.

Для завершения сеанса работы с web-сервисом приложение должно выполнить операцию *Logout*, передав *UserSID* в качестве параметра запроса.

Пример выполнения операций Authenticate и Logout

```
<!DOCTYPE html>
<html>
<head>
```

```
  <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.6.5/angular.min.js"></
script>
  <script src="js/angular-md5.min.js"></script>
</head>
<body>

<div ng-app="ExampleApp">
  <div ng-controller="ExampleController">
    <h3>POST http://localhost:40001/json/Authenticate</h3>
    <b>Request:</b>
    <pre>{{authenticateRequest | json:4 }}</pre>
    <b>Response:</b>
    <pre>{{authenticateResponse | json:4 }}</pre>
    <h3>POST http://localhost:40001/json/Logout</h3>
    <b>Request:</b>
    <pre>{{logoutRequest | json:4 }}</pre>
  </div>
</div>
<script>

angular.module('ExampleApp', ['angular-md5'])
.controller('ExampleController', ['$scope', '$http', 'md5',
  function($scope, $http, md5) {

    $scope.UserSID = '';
    authenticate('admin', 'admin');

    function authenticate(user, password) {
      var PasswordHash =
md5.createHash(md5.createHash(md5.createHash(password).toUpperCase() +
'F593B01C562548C6B7A31B30884BDE53').toUpperCase()).toUpperCase();
      // PasswordHash == '88020F057FE7287D8D57494382356F97'
      $scope.authenticateRequest = {UserName: user, PasswordHash:
PasswordHash};
      $http.post('http://localhost:40001/json/Authenticate',
$scope.authenticateRequest)
          .then(function(response) {
            $scope.authenticateResponse = response.data;
            $scope.UserSID = response.data.UserSID;
            if ($scope.UserSID != '') {
              logout();
            }
          },
          function(response) {
            $scope.authenticateResponse = 'Authentication error';
          });
    }
    function logout() {
```

```
    $scope.logoutRequest = {UserSID: $scope.UserSID};
    $http.post("http://localhost:40001/json/Logout", $scope.logoutRequest);
   }
 }]);

</script>

</body>
</html>
```

Logout

Назначение

Завершение сеанса работы с web-сервисом.

Url: http://localhost:40001/json/Logout

HTTP Метод: POST

Текст запроса Json:

```
{
        "UserSID":"Строковое содержимое"
}
```

Свойства запроса

UserSID
: Уникальный код сеанса пользователя, полученный после выполнения операции *Authenticate*

Пример

См. пример выполнения операции *Authenticate*.

EventGetList

Назначение

Получение журнала событий системы контроля и управления доступом

Url: http://localhost:40001/json/EventGetList

HTTP Метод: POST

Текст запроса Json:

```
{
	"UserSID":"Строковое содержимое",
	"Limit":2147483647,
	"StartToken":9223372036854775807,
	"SubscriptionEnabled":true,
	"IssuedFrom":"\/Date(928138800000+0300)\/"
}
```

Свойства запроса

UserSID
: Уникальный код пользователя, полученный после выполнения операции *Authenticate*

Limit
: Максимальное количество событий в ответе на запрос, соответствующих параметрам *StartToken* и *IssuedFrom*. **Нулевое значение** параметра *Limit* приводит к выдаче событий без ограничения их количества в ответе. **Положительное значение** параметра *Limit* приводит к выдаче событий в порядке возрастания. **Отрицательное значение** параметра *Limit* приводит к выдаче событий в порядке убывания

StartToken
: Числовой идентификатор первого события в запросе. Для поэтапного получения большого количества событий необходимо выполнить несколько запросов, указывая в первом запросе нулевое значение параметра *StartToken*, а в последующих запросах – значение свойства *NextToken* из ответа, полученного на предыдущий запрос

IssuedFrom
: Дата и время самого раннего затребованного события. В ответ не включаются события, зарегистрированные контроллером раньше указанной даты и времени

SubscriptionEnabled
: Признак необходимости включения подписки на события, которые будут добавлены в базу данных после выполнения операции *EventGetList* и дата регистрации которых будет не раньше даты *IssuedFrom*

Текст ответа Json:

```
{
	"UserSID":"Строковое содержимое",
	"NextToken":9223372036854775807,
	"Event":[{
		"Token":9223372036854775807,
		"Card":{
			"Token":9223372036854775807,
			"Name":"Строковое содержимое",
			"Code":2147483647,
```

```
            },
            "CardCode":"Строковое содержимое",
            "Device":{
                  "Token":9223372036854775807,
                  "Name":"Строковое содержимое",
            },
            "EventFilterPassed":true,
            "IconToken":9223372036854775807,
            "Issued":"\/Date(928138800000+0300)\/",
            "Message":{
                  "Token":9223372036854775807,
                  "Name":"Строковое содержимое",
                  "Code":2147483647,
                  "Color":2147483647
            },
            "PhotoToken":9223372036854775807,
            "PhotoverificationFiltersPassed":[true],
            "Sender":{
                  "Token":9223372036854775807,
                  "Name":"Строковое содержимое",
            },
            "User":{
                  "Token":9223372036854775807,
                  "Name":"Строковое содержимое",
            }
      }],
      "EventColumns":["Строковое содержимое"],
      "PhotoverificationColumnCount":2147483647,
      "PhotoverificationColumns":["Строковое содержимое"],
      "PhotoverificationRowCount":2147483647
}
```


Свойства ответа

UserSID

Уникальный код сеанса пользователя, от имени которого была выполнена операция.

NextToken

Идентификатор первого события для последующей операции *EventGetList*. В случае поэтапного получения большого количества события для каждой последующей операции *EventGetList* необходимо передавать значение параметра запроса *StartToken*, равное значению свойства *NextToken* из ответа, полученного в результате выполнения предыдущей операции *EventGetList*

Event

Массив возвращаемых строк таблицы событий

Token	Уникальный код события в базе данных
Card.Token	Уникальный код карточки в базе данных
Card.Name	Название карточки

Card.Code	Код карточки, обнаруженной в базе данных
CardCode	Код карточки, распознанный контроллером
Device.Token	Уникальный код контроллера в базе данных
Device.Name	Название контроллера
EventFilterPassed	Признак соответствия события критерию отображения события в окне событий, который задается администратором для каждого пользователя
IconToken	Уникальный код уменьшенной фотографии пользователя в базе данных
Issued	Дата регистрации события в формате «/Date(zzz±T)/», где zzz – миллисекунды между временной отметкой события и временной отметкой 1970-01-01 00:00:00, T – смещение часового пояса относительно UTC в формате ЧЧММ
Message.Token	Уникальный код типа события в базе данных
Message.Name	Описание типа события, например, «Доступ запрещен»
Message.Code	Код типа события в журнале событий
Message.Color	Цвет, назначенный событиям данного типа в программе-клиенте U-Prox IP
PhotoToken	Уникальный код полноразмерной фотографии пользователя в базе данных
PhotoverificationFilterPassed	Массив признаков соответствия события критериям отображения события в окнах фото-верификации, которые задаются администратором для каждого пользователя
Sender.Token	Уникальный код объекта, с которым связано события – точки доступа, двери, считывателя или контроллера
Sender.Name	Название объекта, с которым связано событие – точки доступа, двери, считывателя или контроллера
User.Token	Уникальный код владельца карточки в базе данных – сотрудника, отдела или посетителя
User.Name	Имя владельца карточки в базе данных – Ф.И.О. сотрудника или посетителя, либо название отдела

EventColumns

Массив идентификаторов колонок таблицы событий, доступных для пользователя, от имени которого выполнена операция EventGetList. Настраивается администратором web-сервиса индивидуально для каждого пользователя

PhotoverificationColumns

Массив идентификаторов колонок таблицы событий, которые необходимо отображать в окне фото-верификации. Настраивается администратором web-сервиса индивидуально для каждого пользователя

PhotoverificationColumnCount

PhotoverificationRowCount

Количество окон фото-верификации по вертикали и по горизонтали на странице пользовательского интерфейса web-сервиса «Охранник»/«Фото-верификация». Настраивается администратором web-сервиса индивидуально для каждого пользователя

Описание

Для получения журнала событий системы необходимо аутентифицировать пользователя с помощью операции Authenticate, а затем выполнить операцию *EventGetList*. В случае поэтапного получения большого количества событий операцию *EventGetList* необходимо выполнить несколько раз, указывая для каждой последующей операции значение параметра запроса *StartToken*, равное значению свойства *NextToken* из ответа, полученного в результате выполнения предыдущей операции.

Для получения событий, которые будут добавлены в базу данных после выполнения операции *EventGetList*, необходимо предварительно установить соединение с signalR-хабом по ссылке http://localhost:40001/signalr, передать ему уникальный код сеанса *UserSID* с помощью вызова метода *Subscribe* и подключить к нему клиентский объект, в котором определен метод *onMessage*.После установки соединения с хабом необходимо выполнить операцию *EventGetList* с параметром запроса *SubscriptionEnabled = true*.

Пример выполнения операции EventGetList и получения новых событий в режиме реального времени (Soft Real-Time)

```
<!DOCTYPE html>
<html>
<head>
  <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.6.5/angular.
min.js"></script>
  <script src="js/angular-md5.min.js"></script>
  <script src="js/jquery.min.js"></script>
  <script src="js/jquery.signalR.min.js"></script>
  <script src="http://localhost:40001/signalr/hubs"></script>
  <style>
    table, th, td {
      border: 1px solid black;
    }
  </style>
</head>
<body>

<div ng-app="exampleApp">
  <div ng-controller="exampleController">
    <h3>POST http://localhost:40001/json/Authenticate</h3>
    <b>Request:</b>
    <pre>{{authenticateRequest | json:4 }}</pre>
    <b>Response:</b>
```

```
    <pre>{{authenticateResponse | json:4 }}</pre>

    <div ng-if="eventGetListRequest">
      <h3>POST http://localhost:40001/json/EventGetList</h3>
      <b>Request:</b>
      <pre>{{eventGetListRequest | json:4 }}</pre>
      <b>Response:</b>
      <pre>{{eventGetListResponse | json:4 }}</pre>

      <div>
        <b>Events:</b>
        <table>
          <tr>
            <td>Card.Token</td>
            <td>Card.Name</td>
            <td>CardCode</td>
            <td>Device.Token</td>
            <td>Device.Name</td>
            <td>Issued</td>
            <td>Message.Code</td>
            <td>Message.Name</td>
            <td>Sender.Token</td>
            <td>Sender.Name</td>
            <td>User.Token</td>
            <td>User.Name</td>
          </tr>
          <tr ng-repeat="x in events">
            <td>{{ x.Card.Token }}</td>
            <td>{{ x.Card.Name }}</td>
            <td>{{ x.CardCode }}</td>
            <td>{{ x.Device.Token }}</td>
            <td>{{ x.Device.Name }}</td>
            <td>{{ x.Date | date:'dd.MM.yy HH:mm:ss' }}</td>
            <td>{{ x.Message.Code }}</td>
            <td>{{ x.Message.Name }}</td>
            <td>{{ x.Sender.Token }}</td>
            <td>{{ x.Sender.Name }}</td>
            <td>{{ x.User.Token }}</td>
            <td>{{ x.User.Name }}</td>
          </tr>
        </table>
      <div>
    </div>

    <div ng-if="newEventGetListRequest">
      <h3>POST http://localhost:40001/json/EventGetList with
subscription for new events</h3>
      <b>Request:</b>
      <pre>{{newEventGetListRequest | json:4 }}</pre>
      <b>Response:</b>
      <pre>{{newEventGetListResponse | json:4 }}</pre>
    </div>
```

```
    <div ng-if="notificationsVisible">
      <div ng-if="timeToStop > 0"><b>Notifications ({{ timeToStop
}} seconds to stop working):</b></div>
      <div ng-if="!timeToStop"><b>Notifications
(stopped):</b></div>
      <table>
        <tr>
          <td>Card.Token</td>
          <td>Card.Name</td>
          <td>CardCode</td>
          <td>Device.Token</td>
          <td>Device.Name</td>
          <td>Issued</td>
          <td>Message.Code</td>
          <td>Message.Name</td>
          <td>Sender.Token</td>
          <td>Sender.Name</td>
          <td>User.Token</td>
          <td>User.Name</td>
        </tr>
        <tr ng-repeat="x in notifications">
          <td>{{ x.Card.Token }}</td>
          <td>{{ x.Card.Name }}</td>
          <td>{{ x.CardCode }}</td>
          <td>{{ x.Device.Token }}</td>
          <td>{{ x.Device.Name }}</td>
          <td>{{ x.Date | date:'dd.MM.yy HH:mm:ss' }}</td>
          <td>{{ x.Message.Code }}</td>
          <td>{{ x.Message.Name }}</td>
          <td>{{ x.Sender.Token }}</td>
          <td>{{ x.Sender.Name }}</td>
          <td>{{ x.User.Token }}</td>
          <td>{{ x.User.Name }}</td>
        </tr>
      </table>
    <div>

    <div ng-if="logoutRequest">
      <h3>POST http://localhost:40001/json/Logout</h3>
      <b>Request:</b>
      <pre>{{logoutRequest | json:4 }}</pre>
    </div>
  </div>
</div>
<script>

var app = angular.module('exampleApp', ['angular-md5']);

app.factory('subscriber', ['$rootScope', '$timeout',
  function($rootScope, $timeout) {
    var Service = {};
    var hub = null;
```

```
    Service.onMessage = function(s) {
      Service.callback(JSON.parse(s));
    };

    Service.disconnect = function() {
      if (!!hub) {
        hub = null;
        $.connection.hub.disconnected(function(){});
        $.connection.hub.stop();
      }
    }

    Service.unsubscribe = function() {
      if (!!hub) {
        hub.server.unsubscribe()
      }
    }

    Service.subscribe = function(callback) {
      if (!hub) {
        Service.callback = callback;
        $.connection.hub.url = 'http://localhost:40001/signalr';
        hub = $.connection.myHub;
        hub.client.onMessage = Service.onMessage;
        $.connection.hub.start().done(function() {
          $rootScope.$apply($rootScope.connected = true);
          hub.server.subscribe($rootScope.UserSID);
        });
        $.connection.hub.reconnected(function() {
          hub.server.subscribe($rootScope.UserSID);
        });
        $.connection.hub.disconnected(function() {
          $rootScope.$apply($rootScope.connected = false);
          if (!!hub) {
            $timeout(function() {
              $.connection.hub.start().done(function() {
                hub.server.subscribe($rootScope.UserSID);
              });
            }, 5000);
          }
        });
      }
    }

    return Service
  }]);

app.controller('exampleController', ['$rootScope', '$scope',
'$http', '$timeout', 'md5', 'subscriber',
  function($rootScope, $scope, $http, $timeout, md5, subscriber) {
    $scope.events = [];
    $scope.notifications = [];
    $scope.UserSID = '';
```

```
    $scope.timeToStop = 30;
    $scope.highestEventToken = 0;
    authenticate('admin', 'admin');

    function authenticate(user, password) {
      var PasswordHash =
md5.createHash(md5.createHash(md5.createHash(password).toUpperCase
() +
'F593B01C562548C6B7A31B30884BDE53').toUpperCase()).toUpperCase();
      $scope.authenticateRequest = {UserName: user, PasswordHash:
PasswordHash};
      $http.post('http://localhost:40001/json/Authenticate',
$scope.authenticateRequest)
           .then(function(response) {
             $scope.authenticateResponse = response.data;
             $scope.UserSID = response.data.UserSID;
             $rootScope.UserSID = response.data.UserSID;
             if ($scope.UserSID != '') {
               eventGetList(2, 0);  // get two events
             }
           },
           function(response) {
             $scope.authenticateResponse = 'Authentication error';
           });
    }

    function eventGetList(count, startToken) {
      $scope.eventGetListRequest = {UserSID: $scope.UserSID,
Limit: -1, StartToken: startToken}; // get one last event
      $http.post('http://localhost:40001/json/EventGetList',
$scope.eventGetListRequest)
           .then(function(response) {
             $scope.eventGetListResponse = response.data;
             response.data.Event.forEach(function(x) {
               if ($scope.highestEventToken < x.Token) {
                 $scope.highestEventToken = x.Token;
               }
             });
             pushIncomingEvents($scope.events,
response.data.Event);
             console.log(count);
             if (count > 1) {
               eventGetList(count - 1, response.data.NextToken);
// get next events
             }
             else {
               newEventGetList(response.data.Event ?
response.data.Event[0].Token + 1 : 0);
             }
           },
           function(response) {
             $scope.eventGetListResponse = 'Event loading error';
           });
```

```
    }

    function newEventGetList(token) {
      subscriber.subscribe(onMessage);
      $scope.newEventGetListRequest = {UserSID: $scope.UserSID,
Limit: 0, StartToken: token, SubscriptionEnabled: true};
      $http.post('http://localhost:40001/json/EventGetList',
$scope.newEventGetListRequest)
           .then(function(response) {
             $scope.notificationsVisible = true;
             $scope.newEventGetListResponse = response.data;
             $timeout(showTimeToStop, 1000);
             $timeout(logout, $scope.timeToStop * 1000);
           },
           function(response) {
             $scope.newEventGetListResponse = 'Event subscription
error';
           });

    }

    function pushIncomingEvents(target, events) {
      events.forEach(function(x) {
        var timeMS = parseInt(x.Issued.substring(6));
        x.Date = new Date(timeMS);
        target.push(x);
      });
    }

    function onMessage(message) {
      if (message.Event) {
        pushIncomingEvents($scope.notifications, message.Event);
        $scope.$apply();
      }
    }

    function logout() {
      subscriber.unsubscribe();
      subscriber.disconnect();
      $scope.logoutRequest = {UserSID: $scope.UserSID};
      $http.post('http://localhost:40001/json/Logout',
$scope.logoutRequest);
    }

    function showTimeToStop() {
      if ($scope.timeToStop) {
        $scope.timeToStop--;
        $timeout(showTimeToStop, 1000);
      }
    }
  }]);

</script>
```

```
</body>
</html>
```