Другие языки:
English • русский

# Содержание



# Поддержка SmartAPI

SmartAPI - API для управления всеми устройствами определенного типа (тип - свет, шторы и т.д.) в проекте. SmartAPI используется в модулях с помощью скрипта.

## Для чего это нужно?

Как известно модули дают возможность приложению i3 lite управлять определенным оборудованием, будь то лампочка Philips HUE, HDL переключатель или какое-то другое устройство. Без SmartAPI модули могли управлять только устройствами для которых их разрабатывали. SmartAPI же позволяет разрабатывать модули, которые работают с конкретными типами устройств. При использовании этого API можно, например, сделать модуль который включает / выключает весь свет в доме.

Рис. Управление с помощью SmartAPI

Принцип работы SmartAPI следующий: SmartAPI ищет устройства нужного типа в проекте и вызывает у всех этих устройств соответствующее действие. На изображении выше SmartAPI вызовет действие "Включить" у всех устройств типа "Свет" в проекте. Таким образом, если в проект добавлены все лампочки, то включится свет во всём доме.

SmartAPI для разработчика модулей можно поделить на 2 части: типизирование модуля (модуль которым управляем) и использование SmartAPI в модуле (модуль который управляет).

## Типизирование модуля

В модуле нужно типизировать сабдевайсы, действия и виртуальные теги. Сабдевайсы необходимо типизировать для того, чтобы SmartAPI понимало, что это за устройство (например свет или кондиционер) и могло им управлять. Модули в которых для устройств не указан тип

не будут управляться с помощью SmartAPI. Пример указания типа в модуле:

```
l_oSubDevice = module.AddSubDevice("Light " + count, HDL_SERVER, false,
IR.SUB_DEVICE_TYPE_LIGHT);
```

Теперь приложение знает, что созданный сабдевайс это устройство для управление светом (константы для всех существующих типов есть во второй вкладке таблицы типов ).

Каждый тип устройства обладает своим набором действий и виртуальных тегов. Например, для света есть следующий набор действий: включить, выключить, переключить, задать яркость, задать теплоту, задать цвет. Также для света есть следующие виртуальные теги: питание (вкл/выкл), яркость, теплота и цвет. Таким образом нужно так же задать тип для действий и виртуальных тегов.

Не обязательно использовать все перечисленные действия и виртуальные теги, устройство имеет набор действий и виртуальных тегов исходя из функционала (если у устройства для управления света нет управления яркостью и теплотой, действия для задания теплоты и цвета просто отсутствуют, также нет смысла в соответствующих тегах).

Пример типизирования действий:

```
in_oSubDevice.AddAction("On", true, turnOn, {SubDevice: in_oSubDevice,
Channel: in_sChannelName}, null, IR.SUB_DEVICE_COMMAND_POWER_ON, null,false);
in_oSubDevice.AddAction("Off", true, turnOff, {SubDevice: in_oSubDevice,
Channel: in_sChannelName}, null, IR.SUB_DEVICE_COMMAND_POWER_OFF, null,
false);
in_oSubDevice.AddAction("Brightness", true, setBrightness, {SubDevice:
in_oSubDevice, Channel: in_sChannelName}, null, IR.SUB_DEVICE_COMMAND_LEVEL,
[{
   Name: "Value",
   Type: IR.ADVANCED_NUMBER,
   Min: ,
   Max: 100
}], false);
in_oSubDevice.AddAction("Power toggle", true, powerToggle, {SubDevice:
in_oSubDevice, Channel: in_sChannelName}, null,
IR.SUB_DEVICE_COMMAND_POWER_TOGGLE, null, true);
in_oSubDevice.AddAction("Power", true, powerOnOff, {SubDevice: in_oSubDevice,
Channel: in_sChannelName}, null, IR.SUB_DEVICE_COMMAND_POWER, null, true);
in_oSubDevice.AddAction("Brightness up", true, brughtnessUp, {SubDevice:
in_oSubDevice, Channel: in_sChannelName}, null,
IR.SUB_DEVICE_COMMAND_LEVEL_UP, null, true);
in_oSubDevice.AddAction("Brightness down", true, brughtnessDown, {SubDevice:
in_oSubDevice, Channel: in_sChannelName}, null,
IR.SUB_DEVICE_COMMAND_LEVEL_DOWN, null, true);
```

Пример типизирования виртуальных тегов:

```
l_oSubDevice.AddVirtualTag("Power", , false, IR.SUB_DEVICE_TAG_POWER, true);
```

```
l_oSubDevice.AddVirtualTag("Level", , false, IR.SUB_DEVICE_TAG_LEVEL, true);
```

Когда вы задали тип для устройства, его действий и виртуальных тегов - значит ваш модуль полностью готов для управления с помощью SmartAPI.

## Использование SmartAPI в модуле

Управление устройствами с помощью SmartAPI реализуется также в модуле. Примеры модулей на основе SmartAPI: модуль для управления всем светом в проекте, модуль для управления всеми шторами в проекте, модуль для управления всем проектом с помощью стороннего сервиса (например Amazon echo) и так далее.

Для того чтобы модуль имел возможность работать со SmartAPI нужно выставить соответствующую галочку при загрузке модуля в iRidium store и редактировании его параметров:

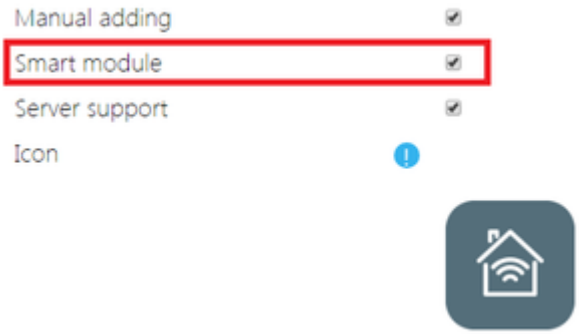


Рис. Модуль для управления через SmartAPI

Пример реализации:

```
function turnOn () {
        var l_aDevices = module.GetSubDevices();
        for (var i = ; i < l_aDevices.length; i++)
                if (l_aDevices[i].SmartID == IR.SUB_DEVICE_TYPE_LIGHT)
                        module.Set(l_aDevices[i],
IR.SUB_DEVICE_COMMAND_POWER_ON);
};

function getVirtualTags () {
        var l_aVirtualTags = [];
        var l_aDevices = module.GetSubDevices();
        for (var i = ; i < l_aDevices.length; i++)
                if (l_aDevices[i].SmartID == IR.SUB_DEVICE_TYPE_LIGHT)
l_aVirtualTags.push(l_aDevices[i].GetVirtualTagBySmartID(IR.SUB_DEVICE_TAG_LE
VEL).Value);
};
```

Функция turnOn из примера включит все устройства типа Light. Функция getVirtualTags заполняет массив значениями яркости со всех устройств типа Light. Все методы для

использования SmartAPI описаны в i3 lite API.

Пример готового модуля с поддержкой SmartAPI (HDL Dimmer): ссылка для загрузки

# SmartAPI

## Методы и свойства

В этом разделе указаны доступные для SmartAPI методы и свойства.

### module.Set

Отправить действие на подустройство

#### Синтаксис

```
module.Set(Subdevice, Command, Value)
```

| Название | Пример | Описание |
|---|---|---|
| `Subdevice` | My_subdevice | type: object<br>Объект подустройства |
| `Command` | POWER в примере метода | type: const<br>Константа команды |
| `Value` | 1 | type: Integer<br>Значение команды (Нужно не для всех команд) |

#### На выходе

-

#### Пример

```
subdevice = module.GetSubDevice ("MySubDevice");
module.Set(subdevice, IR.SUB_DEVICE_COMMAND_POWER, 1);
```

Весь список констант можно найти в разделе типы SmartAPI

### SubDevice.SmartID

Свойство. Получить тип подустройства

#### Синтаксис

```
SubDevice.SmartID
```

| Название | Пример | Описание |
|---|---|---|

-

**На выходе**

Константа типа подустройства

**Пример**

```
var l_aDevices = module.GetSubDevices();
        for (var i = ; i < l_aDevices.length; i++) // Цикл включит все
устройства типа свет в проекте
                if (l_aDevices[i].SmartID == IR.SUB_DEVICE_TYPE_LIGHT)
                        module.Set(l_aDevices[i],
IR.SUB_DEVICE_COMMAND_POWER_ON);
```

Весь список констант можно найти в разделе типы Smart API

**SubDevice.GetVirtualTagBySmartID**

Получить значение виртуального тега подустройства

**Синтаксис**

```
SubDevice.GetVirtualTagBySmartID(Tag const)
```

| Название | Пример | Описание |
|---|---|---|
| Tag const | IR.SUB_DEVICE_TAG_LEVEL | type: const<br>Тип тега |

**На выходе**

Значение тега указанного типа

**Пример**

```
        var l_aVirtualTags = [];
        var l_aDevices = module.GetSubDevices();
        for (var i = ; i < l_aDevices.length; i++) // Цикл запишет значения
тегов яркости всех устройств типа свет в проекте в массив
                if (l_aDevices[i].SmartID == IR.SUB_DEVICE_TYPE_LIGHT)
l_aVirtualTags.push(l_aDevices[i].GetVirtualTagBySmartID(IR.SUB_DEVICE_TAG_LE
VEL).Value);
```

Весь список констант можно найти в разделе типы SmartAPI

## Типы

Поддерживаемые типы SmartAPI указаны в этой таблице: ссылка