



**МПК-ПЕРЕСВЕТ**

**Программа для ЭВМ «МПК-Пересвет»**

**Руководство пользователя**

Москва 2025

## ГЛОССАРИЙ

Термин	Определение
Модель технического объекта	Совокупность статической и динамической моделей объекта.
Статическая модель объекта	Иерархия сущностей. Также в иерархию включаются сущности, описывающие информационную систему. В простейшем случае статическая модель может представлять собой просто линейный список тегов (параметров объектов).
Динамическая модель	Совокупность методов, реализуемых на каких-либо языках программирования. В виде методов реализуются бизнес-процессы, протекающие на моделируемом объекте, а также выполняются расчёты значений рассчитываемых тегов. Выполнение методов инициируется при возникновении событий в модели.
Сущность	Основные сущности статической модели технического объекта: теги, объекты, тревоги, методы, расписания, константы. Дополнительные сущности, описывающие информационную систему: хранилище данных, источник данных. Также есть возможность создавать дополнительные сущности, самостоятельно реализуя логику работы с ними.
Тег	<p>Параметр объекта. Например: температура, давление, сила тока и т.д. Значения тегов хранятся в том или ином хранилище данных. Каждое значение тега обязательно имеет метку времени. Кроме того, может иметь показатель качества значения.</p> <p>Тег в иерархии представляется узлом, дочерним по отношению к какому-либо объекту.</p> <p>Поддерживаемые типы значений тегов: целый, вещественный, строковый, json.</p>
Объект	Основная сущность при моделировании технических объектов. Объектом может быть: предприятие, цех, участок, технологическая линия, агрегат, датчик и т.д. Каждый объект может содержать любое количество объектов-потомков. Также объект может содержать любое количество тегов.
Расписание	Заданная последовательность, определяющая моменты времени, в которые будут выполняться те или иные задачи. Задачами могут быть: запуск расчёта вычисляемых тегов, запуск на исполнение каких-либо методов (к примеру, генерация отчётов и рассылка их по почте).
Событие	В платформе поддерживаются три основных типа событий: изменение тега, возникновение тревоги, событие расписания. К этим событиям можно привязывать выполнение различных методов, тем самым «оживляя» модель технического объекта. События являются инициаторами действий.

Тревога	<p>Это состояние тега, которое может возникнуть при изменении тега и при выполнении некоторых заданных условий. Есть четыре вида стандартных тревог: LoLo, Lo, Hi, HiHi.</p> <p>В «МПК-Пересвет», в отличие от типовых тревог LoLo-Lo-Hi-HiHi, есть возможность создать любое их количество. Более того, возможно создавать сложные тревоги, возникновение которых учитывает значение других тегов. Это достигается тем, что условием возникновения тревоги может быть указано не только какое-то значение тега, а результат расчёта метода, который может реализовывать любую сложную логику.</p>
Метод	<p>Программный код, реализующий логику поведения моделируемого объекта. Методы вызываются на исполнение возникающими событиями и используются для: расчёта значений вычисляемых тегов, определения факта возникновения/пропадания тревоги, вызова внешних процессов и т.д.</p>
Хранилище данных	<p>База данных, в которой хранятся исторические данные (значения тегов). Платформа может поддерживать одновременно несколько хранилищ данных разных типов. Рекомендуемое хранилище данных для систем промышленной автоматизации - PostgreSQL. Также есть драйвер для VictoriMetrics. Возможно написание драйверов для любых других типов хранилищ данных.</p>
Коннектор	<p>Программа, обеспечивающая сбор данных с устройства по определённому протоколу. Обычно для каждого протокола создаётся отдельный коннектор.</p>

## Содержание

Назначение программы для ЭВМ «МПК-Пересвет».....	6
Что должен знать пользователь.....	6
Исторические данные.....	6
Правила работы с данными.....	6
Коды качества.....	7
Получение данных.....	8
Пример массива данных.....	8
Формат запроса на получение данных.....	8
Формат ответа на запрос на получение данных.....	10
Получение текущего значения тэга.....	10
Метки времени в будущем и prsStep = false.....	11
Метки времени в будущем, prsStep = true.....	12
Ключ start.....	13
prsStep = false.....	13
prsStep = true.....	14
Начало периода в запросе - перед любой меткой времени в базе.....	15
Начало периода в запросе - после любой метки времени в базе.....	16
Ключ finish.....	17
finish = произвольной метке времени.....	17
Ключ finish установлен перед любой существующей в базе меткой времени.....	18
Ключ finish равен текущей метке времени.....	18
Ключи start и finish.....	19
start и finish внутри существующих в базе меток.....	19
start выходит за пределы меток.....	20
start и finish вне меток времени.....	21
Ключ count.....	22
Ключи count и finish.....	22
Ключи count и start.....	25
Ключ timeStep.....	28
timeStep и count.....	28
timeStep, count и finish.....	30
timeStep, count и start.....	30
timeStep, start и finish.....	32
Ключ maxCount.....	33
Ключ actual.....	33
start и actual.....	33
start, count, actual.....	34
actual и finish.....	36
finish, count, actual.....	37
start, finish, actual.....	38
start, finish, count, actual.....	39
start, finish, maxCount, actual.....	40
Ключ value.....	42
value и actual.....	43
null в истории данных.....	44

---

Примеры работы с МПК-Пересвет.....	50
Простой пример с одним объектом.....	50
Объект.....	50
Теги.....	52
Запись/чтение данных.....	55
Расчётный метод.....	58
Запуск расчётного метода.....	64
Проверим расчёт.....	64
Отображение данных в Grafana.....	67

## Назначение программы для ЭВМ «МПК-Пересвет»

Программа для ЭВМ «МПК-Пересвет» (далее по тексту – платформа или система) предназначена для создания моделей технических объектов и информационных систем.

Через создание моделей объектов появляется возможность мониторинга и управления ими.

Примеры моделируемых объектов: агрегат, технологическая линия, цех, предприятие, частный дом, здание, ТЭЦ, и т. д.

## Что должен знать пользователь

Пользователь системы должен понимать принципы работы с историческими данными в платформе МПК-Пересвет.

Экраны для пользователя создаёт администратор, соответственно, логика работы с системой зависит от логики работы экранов, в общем случае — от логики модели, созданной администратором на базе МПК-Пересвет.

## Исторические данные

Одна из главных задач Платформы - обеспечение клиентов сохранёнными историческими данными. Клиенты получают данные с помощью запроса GET по адресу /v1/data/.

Эта команда будет подробно объяснена ниже.

## Правила работы с данными

Важно:

При работе с историческими данными запомним следующие правила:

1. Для правильной работы с историческими данными все сервера, устройства, клиенты должны быть синхронизированы по времени.
2. Если в тэге нет значений с метками времени из будущего, то последнее сохранённое на текущий момент времени в тэге значение является текущим значением тэга.
  1. Если значения тэга числовые:
  2. Если атрибут тэга `prsStep` равен `true`, тогда значения тэга между двумя метками времени не интерполируются.
  3. Если атрибут тэга `prsStep` равен `false` (по умолчанию), тогда значения тэга между двумя метками времени вычисляются с помощью линейной интерполяции.

4. Программы, предоставляющие данные пользователям, ответственны за верную интерпретацию атрибута prsStep у тэга. К примеру, тренд должен правильно показывать значения тэга между соседними метками времени.
3. Все метки времени внутри платформы и в исторической базе данных хранятся и обрабатываются как целочисленные данные. Каждая метка времени - это количество микросекунд, начиная с 01-01-1970 00:00:00+00:00. Пример: 1000000 = 01-01-1970 00:00:01+00:00
4. Если клиент посылает в платформу метку времени как строку в формате ISO 8601 и не указывает временную зону, то Платформа устанавливает для такой метки времени зону по умолчанию UTC (+00:00).
5. Клиенты могут записывать значения тэгов с метками времени из будущего.
6. Ответ на запрос GET /v1/data/ содержит массив значений тэга, отсортированный по метке времени в возрастающем порядке.
7. Если тэг содержит несколько значений на одну и ту же метку времени, то текущим значением тэга на эту метку времени считается записанное последним.
8. Каждое значение в тэге содержит код качества q. Если q = 0 или null, то значению тэга можно доверять. Иначе значение тэга ошибочно, код q определяет ошибку. См. коды качества.
9. Следующие правила касаются ключей count, start, finish и timeStep в запросе GET /v1/data/.
10. Комбинации ключей start, finish и timeStep совместно с ключом count определяют границы периода для возвращаемых данных.
11. Ключ count определяет количество возвращаемых данных. Учитывается общее количество: и реально записанные в базе данных, и интерполированные значения тэгов.
12. В случае отсутствия ключа finish его значение принимается равным текущему моменту времени. Это правило по умолчанию отсекает данные, записанные с метками времени из будущего.

## Коды качества

У каждого записанного в базу значения тэга есть так называемый «код качества», который возвращается в ключе q.

Код качества - это, другими словами, код ошибки данных.

Код	Описание
0   null	Обычное значение. Данные «хорошие».
100	Разорвана связь между платформой и коннектором, записывающим данные в тэг. Используется совместно со значением тэга <code>null</code> .
101	Платформа завершила работу. Используется совместно со значением тэга <code>null</code> и записывается в процессе завершения платформой работы.
102	Разорвана связь между коннектором и поставщиком данных. Используется совместно со значением тэга <code>null</code> .

## Получение данных

Подробно рассмотрим примеры получения исторических данных.

### Пример массива данных

Допустим, у нас есть тэг «Tag1» у объекта «Object1». Идентификатор тега — «1500c712-726a-103e-9264-a5021ec2dae1». Атрибут «prsTagValueTypeCode» этого тэга равен 1 и это значит, что тэг хранит вещественные значения. Исторические данные этого тэга:

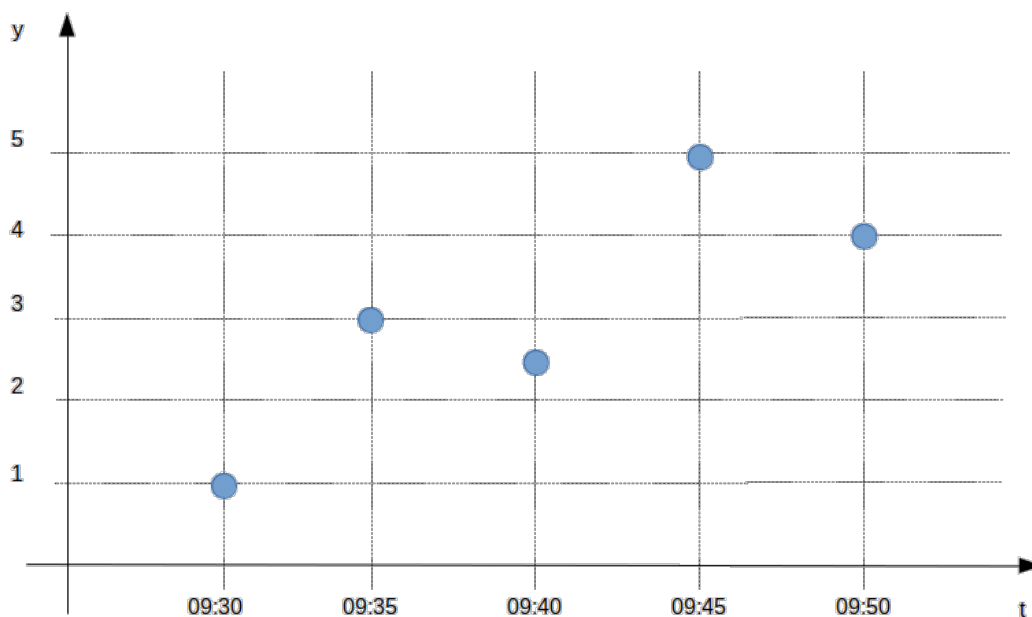


Рисунок 1: Данные тэга

---

Для простоты будем оперировать только часами и минутами в качестве меток времени.

---

Итак, исторические данные тэга Tag1:

```
[  
  [1, "09:30"],  
  [3, "09:35"],  
  [2.5, "09:40"],  
  [5, "09:45"],  
  [4, "09:50"]  
]
```

## Формат запроса на получение данных

Пример:

```
{
  "tagId": ["id первого тэга", "id второго тэга"],
  "start": "2018-12-09 12:00:00+03:00",
  "finish": "2018-12-09 14:00:00+03:00",
  "count": 8,
  "timeStep": 60000000,
  "maxCount": 700,
  "actual": false,
  "format": true,
  "value": 10
}
```

**tagId** (str или массив из str), обязательный — тэг(и) для которых необходимо получить данные.

**start** (int, str), необязательный — метка времени начала периода для получения данных (see). Если start - строка, то ее значение обрабатывается в соответствии со стандартом ISO 8601.

Возможные значения:

- «2018-12-20 00:00:00+03:00».
- «2018-12-20 00:00:00» - временная зона для этого значения будет установлена в UTC (00:00).
- 1544662830000000 значение равно: «2018-12-13 01:00:30+00:00».
- «01:00» предположим, что текущая дата на сервере платформы - 2018-12-20, тогда «01:00» будет преобразовано в «2018-12-20 01:00:00+00:00».
- «01:00,5+03:00» равно (оставим предыдущее предположение о дате) «2018-12-20 01:00:30+3:00».
- другие примеры указания времени можно посмотреть на странице стандарта ISO 8601.

**finish** (int, str), необязательный — метка времени конца периода, соответствует тем же правилам, что и ключ start.

**count** (int), необязательный — количество значений, ожидаемых в ответе.

**timeStep** (int), необязательный — промежуток времени между соседними возвращаемыми значениями тэга

**maxCount** (int), необязательный — ключ используется, в основном, виджетами и принимает значение ширины виджета в пикселях. Таким образом, платформа знает, что виджет не может показать больше значений, чем указано в maxCount. В этом случае платформа возвращает не больше значений, чем указано в maxCount.

Ключ maxCount предотвращает излишнюю загрузку платформы при работе с большими массивами данных.

**format** (любой тип и значение), необязательный — если этот ключ присутствует и не равен None, тогда метки времени возвращаются в виде строк в формате ISO 8601, часовая зона - зона сервера, на котором работает платформа.

**actual** (bool), необязательный — если этот ключ присутствует и установлен в true, тогда в ответе на запрос присутствуют только реальные записанные базу данных значения тэга.

**value** (any), необязательный — фильтр значений тега.

## Формат ответа на запрос на получение данных

Пример:

```
{
  "data": [
    {
      "exceed": false,
      "tagId": "id тега",
      "data": [
        [1, "2018-12-31T00:00:00+03:00", null],
        [120, "2018-12-31T00:01:00+03:00", null]
      ]
    }
  ]
}
```

Если запрос выполнен без ошибок, ответ всегда содержит один ключ — data, который является массивом элементов типа json, каждый элемент — данные по одному тегу из запроса.

Ключи элемента из массива:

**exceed** (bool) — флаг того, что данных в архиве больше, чем запрошено; логика работы флага: если в запросе указан ключ maxCount, то в ответе устанавливается флаг exceed; если этот флаг = true, то это означает, что в архиве содержится больше значений, чем указано в запросе в ключе maxCount, поэтому возвращённые значения тега усреднены и лучший выход - изменить запрос, запросив меньшее количество значений тега; если в запросе нет ключа maxCount, то ключ exceed в ответе отсутствует;

**tagId** (str) — идентификатор тега;

**data** (массив массивов) — массив значений, попадающих в запрошенный период; каждый элемент массива - в свою очередь, тоже массив и состоит из трёх элементов: [y, x, q],

где:

y - значение тега;

x - метка времени, с которой записано значение тега;

q - код качества значения.

Ниже — примеры запросов.

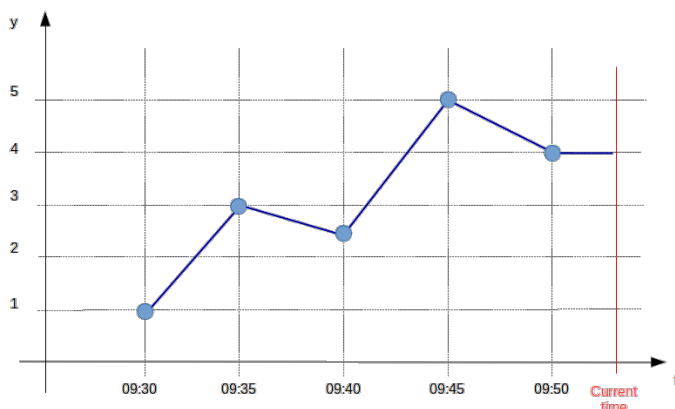
## Получение текущего значения тэга

Самый простой запрос на получение данных выглядит так:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec"
}
```

Запрос выше показывает, как можно получить текущее значение одного тэга.

Допустим, текущий момент времени — 09:53:



Ответом будет:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [4, 1545288780000000, null]
      ]
    }
  ]
}
```

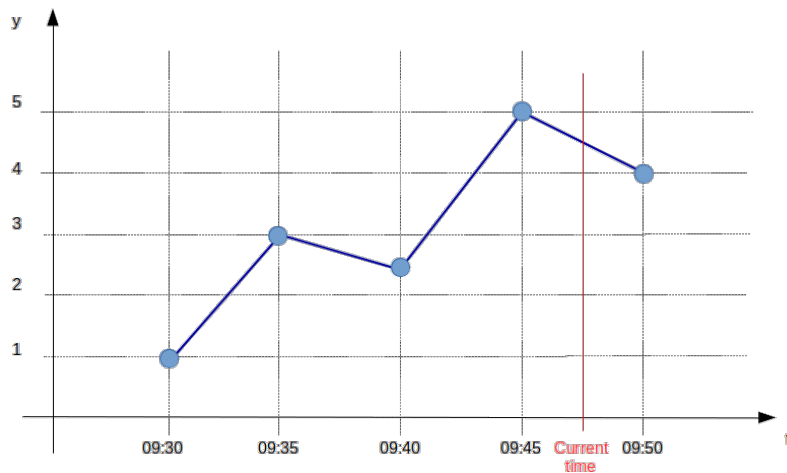
В соответствии с правилом 2 текущее значение тэга - 4. Метка времени (1545288780000000) соответствует текущему моменту времени. Такой ответ будет идентичен для тэгов с атрибутом `prsStep`, равным `true` и `false`.

*Для простоты понимания меток времени во всех следующих запросах будем использовать ключ `format`.*

*Опять же для простоты исключим из всех дальнейших меток времени дату.*

## Метки времени в будущем и `prsStep = false`

Допустим, текущий момент времени 09:47:30. Это значит, что мы имеем одно значение тэга, сохранённое с меткой времени в будущем. Пусть `prsStep = false`, то есть мы должны интерполировать значения тэгов.



Тогда запрос

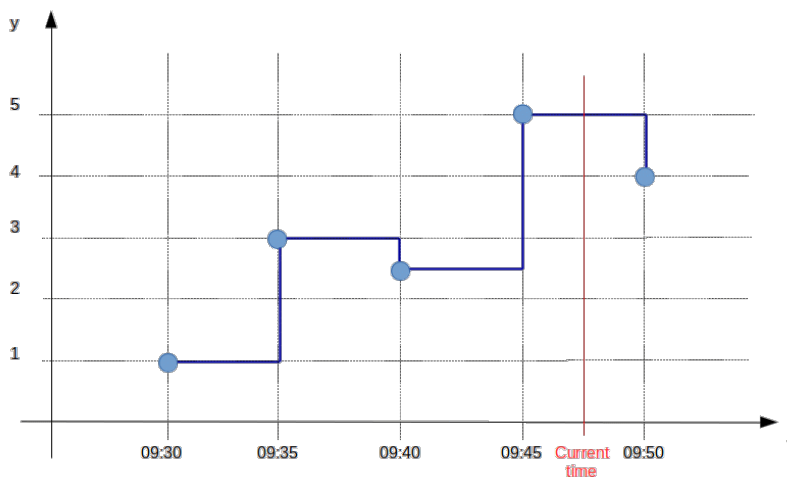
```
{  
  "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",  
  "format": true  
}
```

Вернёт:

```
{  
  "data": [  
    {  
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",  
      "data": [  
        [4.5, "09:47:30", null]  
      ]  
    }  
  ]  
}
```

## Метки времени в будущем, prsStep = true

Этот пример отличается от предыдущего тем, что prsStep = true.



Тогда запрос

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
  "format": true
}
```

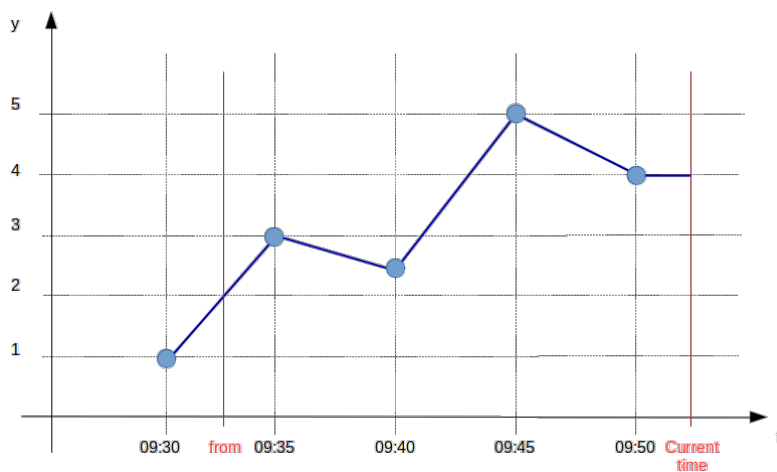
Вернёт:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [5, "09:47:30", null]
      ]
    }
  ]
}
```

## Ключ start

### prsStep = false

Добавим к нашему запросу ключ start. Ключ start устанавливает начало периода для получения данных. Пусть start = 09:32:30 и prsStep = false.



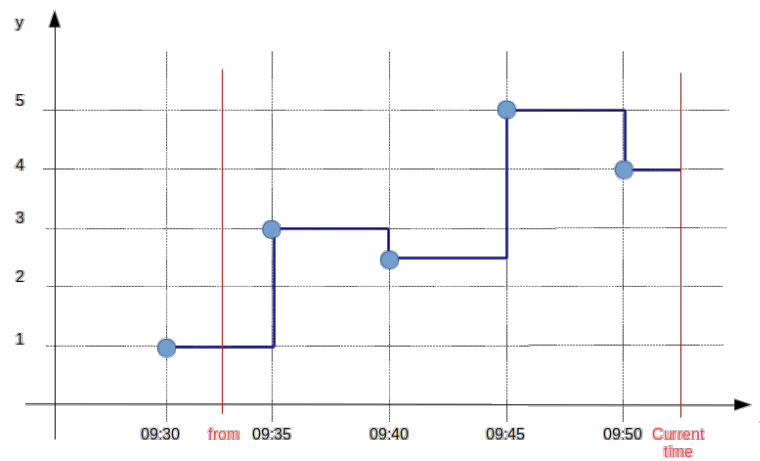
Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
  "format": true,
  "start": "09:32:30"
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [2, "09:32:30", null],
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null],
        [4, "09:53:00", null]
      ]
    }
  ]
}
```

**prsStep = true**



Запрос:

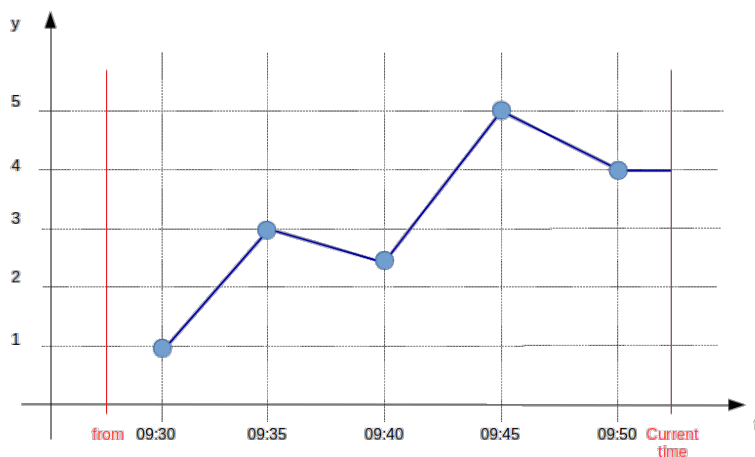
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
  "format": true,
  "start": "09:32:30"
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [1, "09:32:30", null],
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null],
        [4, "09:53:00", null]
      ]
    }
  ]
}
```

### Начало периода в запросе - перед любой меткой времени в базе

Пусть start = 09:27:30.



Запрос:

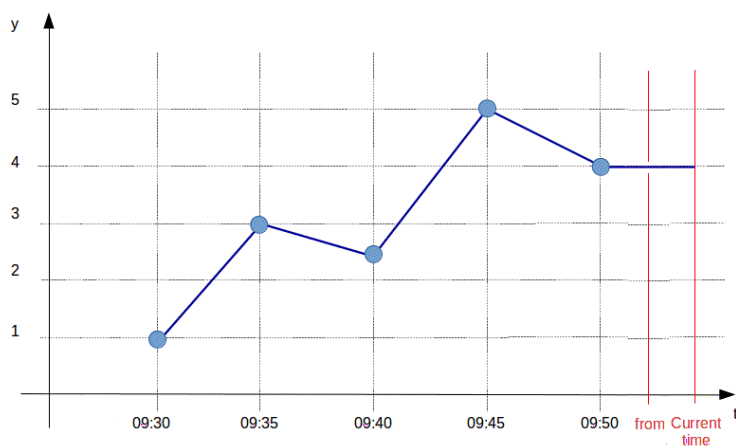
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:27:30"
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [null, "09:27:30", null],
        [1, "09:30:00", null],
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null],
        [4, "09:53:00", null]
      ]
    }
  ]
}
```

### Начало периода в запросе - после любой метки времени в базе

Пусть start = 09:52:30.



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:52:30"
}
```

Возвращается последнее значение в базе как на метку start, так и на текущую метку времени.

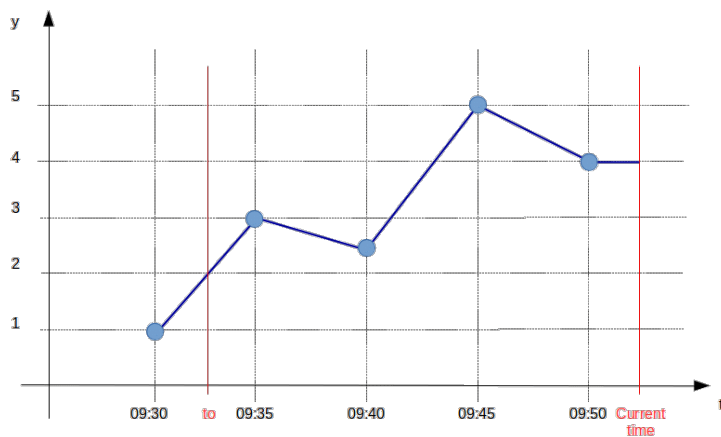
Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [4, "09:52:30", null],
        [4, "09:55:00", null]
      ]
    }
  ]
}
```

## Ключ finish

### finish = произвольной метке времени

Устанавливая ключ finish, мы ограничиваем конец периода получения данных.



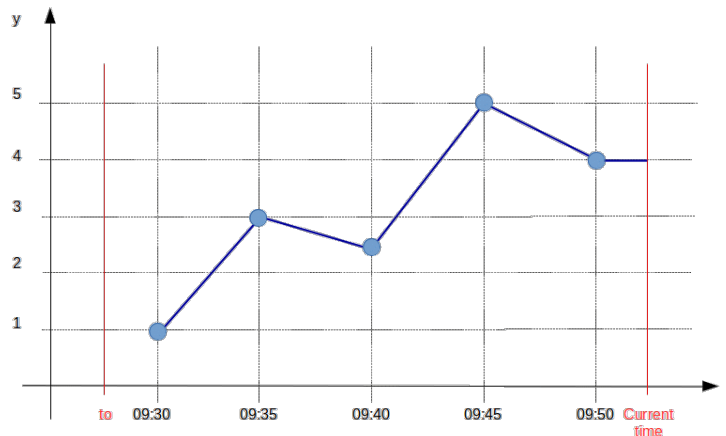
Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "finish": "09:32:30"
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [2, "09:32:30", null]
      ]
    }
  ]
}
```

## Ключ finish установлен перед любой существующей в базе меткой времени



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "finish": "09:27:30"
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        ["09:27:30", "y": null]
      ]
    }
  ]
}
```

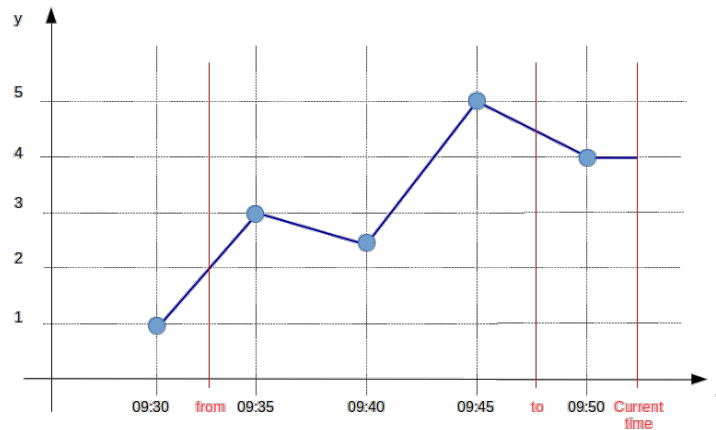
## Ключ finish равен текущей метке времени

Если мы установим ключ finish на текущую метку времени, то запрос будет в точности таким же, как в запросе «Получение текущего значения тэга».

## Ключи start и finish

Ниже — несколько примеров одновременно установленных ключей start и finish.

### start и finish внутри существующих в базе меток



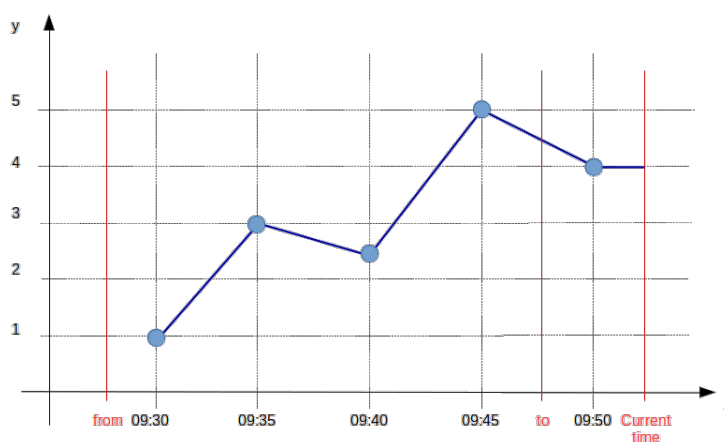
Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "finish": "09:47:30"
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [2, "09:32:30", null],
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4.5, "09:47:30", null]
      ]
    }
  ]
}
```

### start выходит за пределы меток



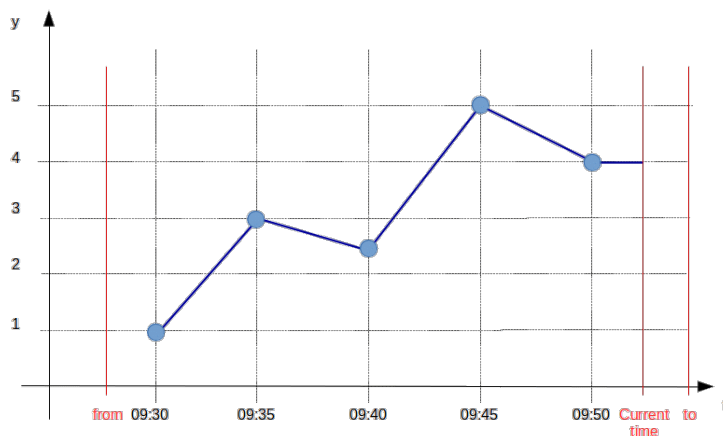
### Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:27:30",
  "finish": "09:47:30"
}
```

### Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [null, "09:27:30", null],
        [1, "09:30:00", null],
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4.5, "09:47:30", null]
      ]
    }
  ]
}
```

### start и finish вне меток времени



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:27:30",
  "finish": "09:55:00"
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [null, "09:27:30", null],
        [1, "09:30:00", null],
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null],
        [4, "09:55:00", null]
      ]
    }
  ]
}
```

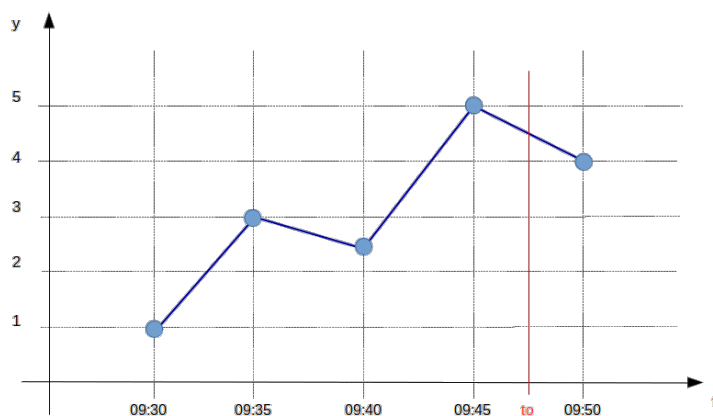
## Ключ count

Ключ count определяет запрашиваемое количество данных.

В соответствии с правилом 10 ключ finish равен текущей метке времени в случае, если установлен только ключ count.

## Ключи count и finish

Предположим, что мы установили ключ finish на момент времени, не существующий в базе данных:



Запрос:

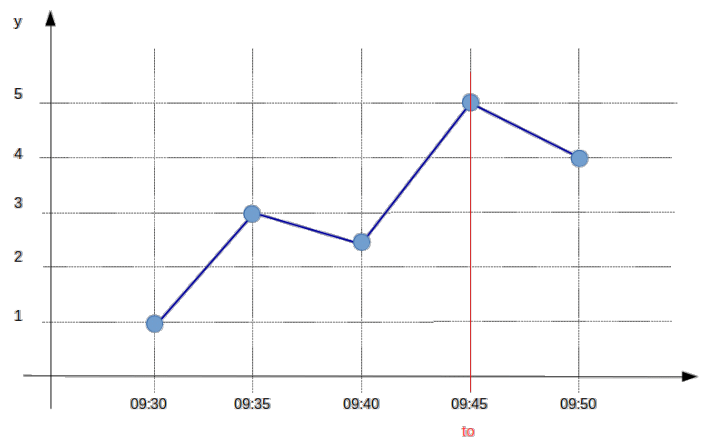
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "finish": "09:47:30",
  "count": 3
}
```

Этот запрос вернёт два сохранённых значения и одно интерполированное.

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4.5, "09:47:30", null]
      ]
    }
  ]
}
```

Следующий пример показывает, что мы получим в случае, если ключ `finish` равен существующей в базе метке времени:



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "finish": "09:45:00",
  "count": 3
}
```

Запрос вернёт три сохранённых значения.

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null]
      ]
    }
  ]
}
```

Запрос:

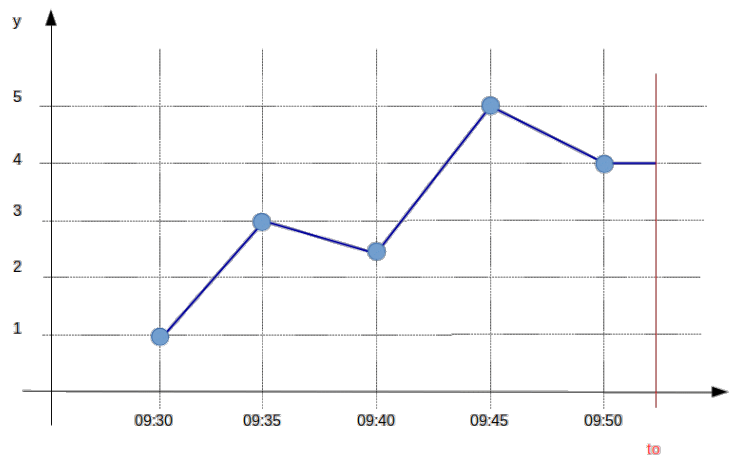
Ключ count превышает количество существующих значений (основываемся на предыдущей картинке).

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
  "format": true,
  "finish": "09:45:00",
  "count": 5
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [1, "09:30:00", null],
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null]
      ]
    }
  ]
}
```

Ключ finish превышает последнюю существующую метку времени:



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "finish": "09:52:30",
  "count": 3
}
```

Будут возвращены два сохранённых и последнее существующее значение на метку времени `finish`:

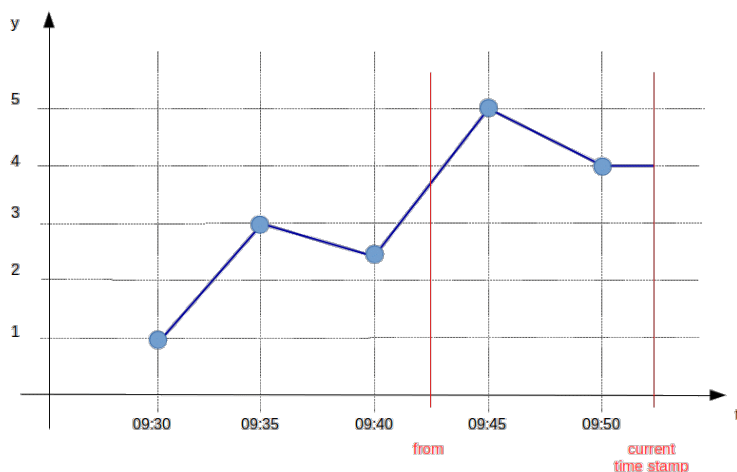
Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [5, "09:45:00", null],
        [4, "09:50:00", null],
        [4, "09:52:30", null]
      ]
    }
  ]
}
```

## Ключи `count` и `start`

Совместное использование ключей `count` и `start` похоже на случай совместного использования ключей `finish` и `count` за исключением того, что мы ограничиваем не конец периода, а начало.

Несколько следующих примеров основываются на следующей картинке:



Запрос `count = 1`:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:42:30",
  "count": 1
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3.3, "09:42:30", null]
      ]
    }
  ]
}
```

Запрос count = 2:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:42:30",
  "count": 2
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3.3, "09:42:30", null],
        [5, "09:45:00", null]
      ]
    }
  ]
}
```

Запрос count = 3:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:42:30",
  "count": 3
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3.3, "09:42:30", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null]
      ]
    }
  ]
}
```

Запрос count = 4:

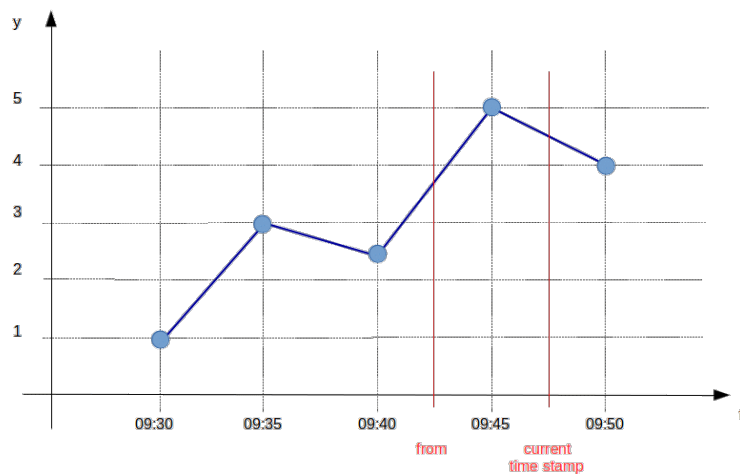
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:42:30",
  "count": 4
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3.3, "09:42:30", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null],
        [4, "09:52:30", null]
      ]
    }
  ]
}
```

Запрос count = 5: ответ будет в точности таким же, как предыдущий.

Давайте переместим текущую метку времени:



Запрос count = 3:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:42:30",
  "count": 3
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3.3, "09:42:30", null],
        [5, "09:45:00", null],
        [4.5, "09:47:30", null]
      ]
    }
  ]
}
```

Запрос count = 4: ответ будет таким же, как предыдущий в связи с правилом 10, вне зависимости от того, что существует значение с меткой времени в будущем (09:50:00).

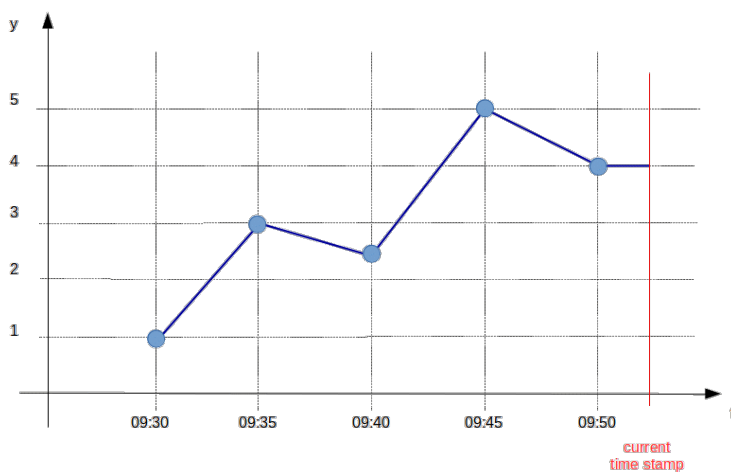
## Ключ timeStep

Ключ timeStep устанавливает временной период между соседними возвращаемыми значениями. Используйте этот ключ совместно с ключом count.

Значения ключа timeStep измеряется в микросекундах.

## timeStep и count

Как определено в правиле 2, значение по умолчанию ключа finish - текущая метка времени.



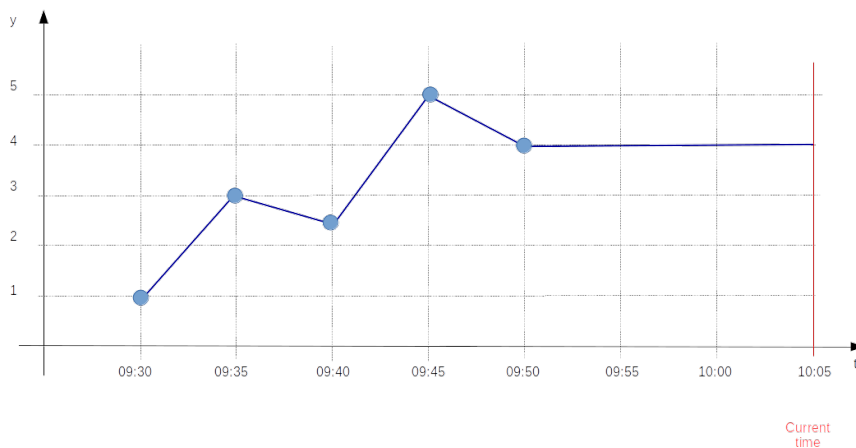
Запрос (timeStep равен 5 минутам):

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "timeStep": 300000000,
  "count": 3
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3.7, "09:42:30", null],
        [4.5, "09:47:30", null],
        [4, "09:52:30", null]
      ]
    }
  ]
}
```

Теперь рассмотрим тот же запрос, но выполненный в момент времени 10:05:



Запрос (timeStep равен 5 минутам):

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "timeStep": 300000000,
  "count": 3
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [4, "09:55:00", null],
        [4, "10:00:00", null],
        [4, "10:05:00", null]
      ]
    }
  ]
}
```

Запрос (count = 6):

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "timeStep": 300000000,
  "count": 6
}
```

Ответ:

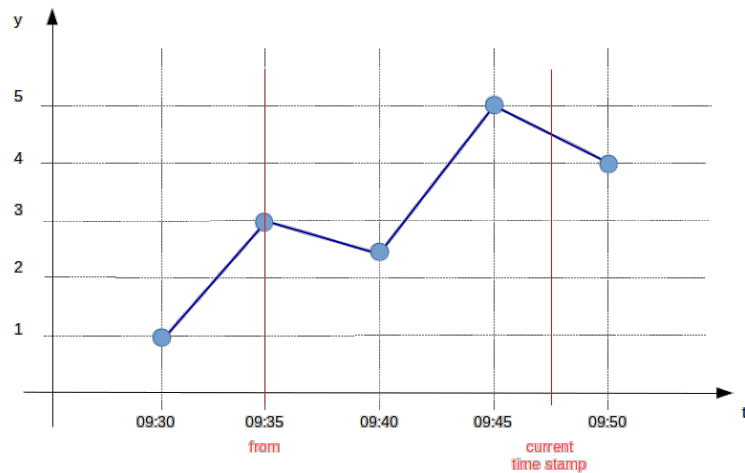
```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [null, "09:27:30", null],
        [2, "09:32:30", null],
        [2.725, "09:37:30", null],
        [3.3, "09:42:30", null],
        [4.5, "09:47:30", null],
        [4, "09:52:30", null]
      ]
    }
  ]
}
```

## timeStep, count и finish

Поведение запросов с ключами timeStep, count и finish такое же, как в предыдущих примерх, за исключением того, что ключ finish явно задаёт окончание периода.

## timeStep, count и start

В этом случае ключ finish вычисляется в соответствии с правилом 12.



Запрос (timeStep равен 5 минутам):

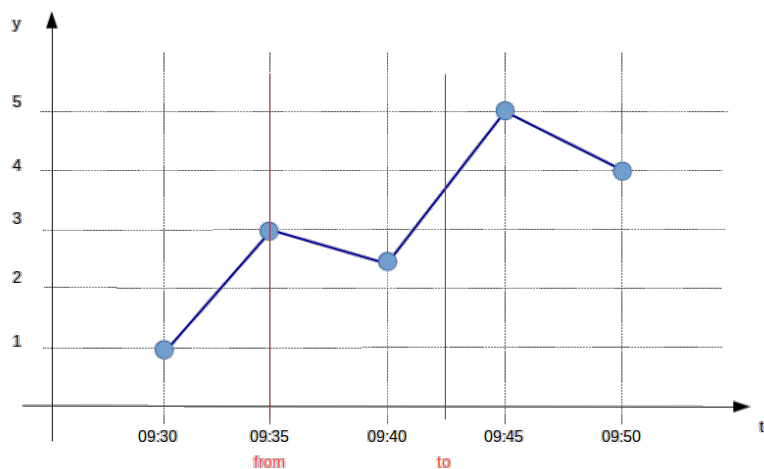
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:35:00",
  "timeStep": 300000000,
  "count": 4
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [4.5, "09:45:00", null]
      ]
    }
  ]
}
```

Несмотря на то, что ключ count равен 4, возвращено только 3 значения, т.к. еще одно значение, имеющееся в базе, относится к будущему.

## timeStep, start и finish



Запрос (timeStep равен 5 минутам):

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:35:00",
  "finish": "09:42:30",
  "timeStep": 300000000
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null]
      ]
    }
  ]
}
```

Запрос (timeStep равен 10 минутам):

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:35:00",
  "finish": "09:42:30",
  "timeStep": 600000000
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null]
      ]
    }
  ]
}
```

## Ключ `maxCount`

Этот ключ используется клиентами для того, чтобы информировать платформу, какое максимальное количество значений тэга должно быть возвращено.

Представим тренд на экране. Его ширина - 800 пикселей. Это значит, что тренд может отрисовать не более 800 значений по оси x.

Тренд устанавливает ключ `maxCount` в 800, при любом запросе на получение данных. Если в хранилище больше, чем 800 значений за выбранный период, платформа установит в ответе ключ `excess` в `true` и вернёт только 800 значений.

---

*`maxCount` имеет приоритет над `count`.*

---

---

*Использование ключа `maxCount` предотвращает крах платформы и повышает скорость работы. Установка этого ключа не обязательна, но рекомендуется.*

---

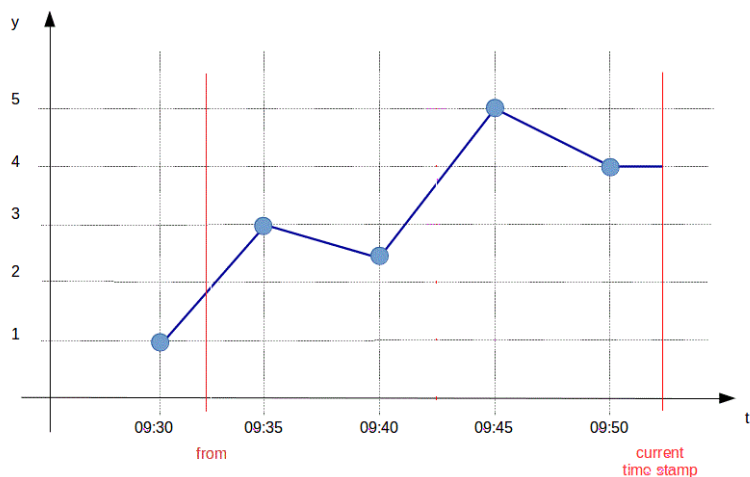
## Ключ `actual`

Если ключ `actual` установлен в `true`, выводятся только данные, находящиеся в базе данных без интерполяции/присвоения на границах диапазона. По умолчанию ключ `actual` установлен в `false`. Если ключ `actual` установлен в `true`, ключ `timeStep` не учитывается.

Ниже представлены случаи использования ключа `actual` с различными комбинациями других ключей.

### **start и actual**

В этом случае возвращаются данные от метки времени, заданной `start`, до текущего момента времени. Если на метку `start` и/или на текущий момент времени данных нет, на эти метки времени ничего не возвращается.



Запрос:

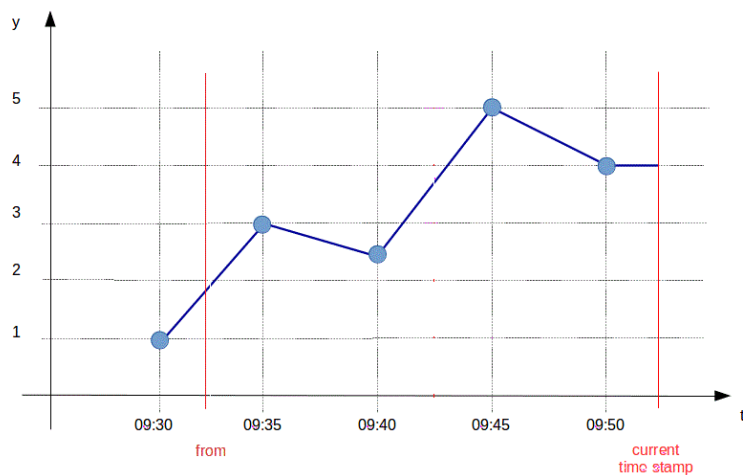
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:00",
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null]
      ]
    }
  ]
}
```

### start, count, actual

Возвращаются данные в количестве, заданном в count. Если данных в базе меньше, чем задано в count, возвращаются только имеющиеся данные.



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:00",
  "count": 3,
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null]
      ]
    }
  ]
}
```

Запрос:

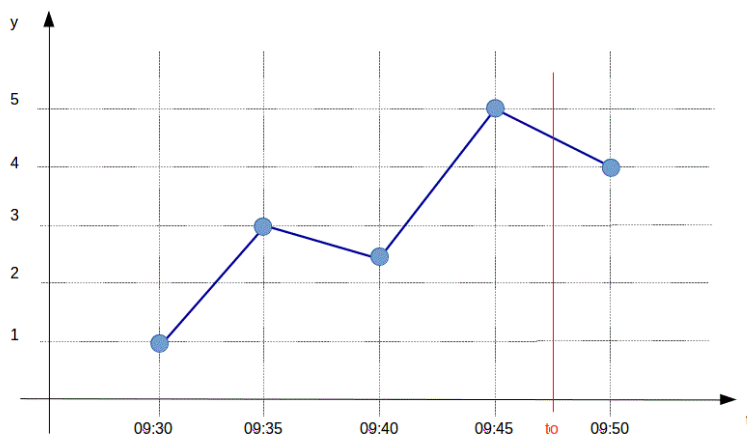
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:00",
  "count": 10,
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null]
      ]
    }
  ]
}
```

## actual и finish

В этом случае возвращается значение на метку времени finish или последнее имеющееся значение перед меткой времени finish с соответствующей меткой времени, если значения на метку времени finish нет.



Запрос:

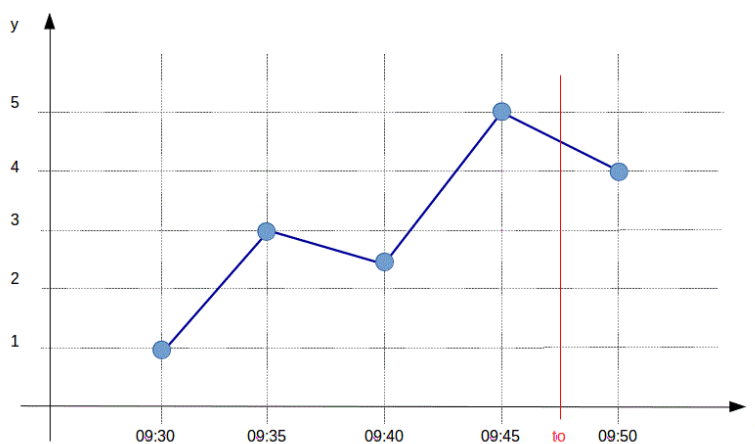
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "finish": "09:47:30",
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [5, "09:45:00", null]
      ]
    }
  ]
}
```

## finish, count, actual

Возвращаются данные в количестве, заданном в count, от метки времени finish. Если данных в базе меньше, чем задано в count, возвращаются только имеющиеся данные.



Запрос:

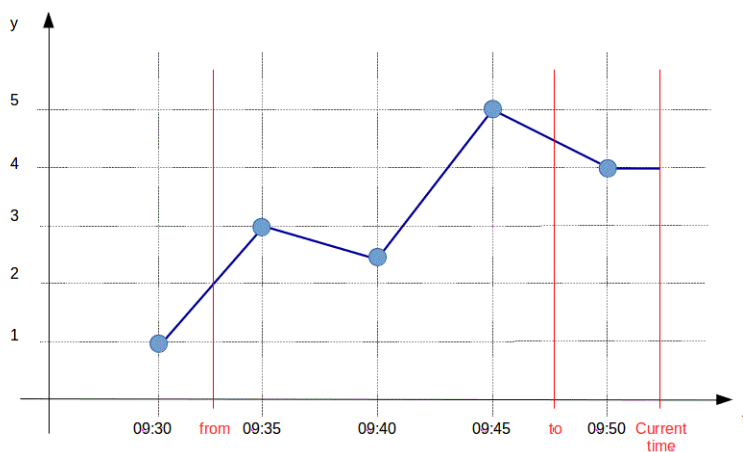
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "finish": "09:47:30",
  "count": 3,
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null]
      ]
    }
  ]
}
```

## start, finish, actual

В этом случае возвращаются имеющиеся в базе данные от метки времени start до метки времени finish.



Запрос:

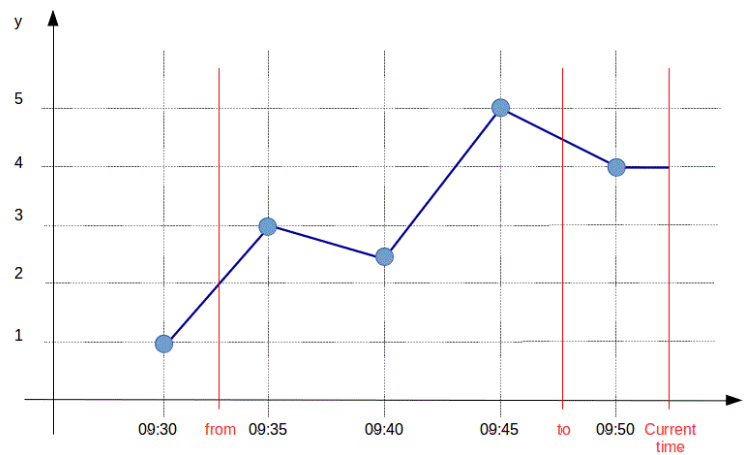
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "finish": "09:47:30",
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null]
      ]
    }
  ]
}
```

### start, finish, count, actual

Возвращаются данные в количестве, заданном в count. Если данных в базе меньше, чем задано в count, возвращаются только имеющиеся данные.



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "finish": "09:47:30",
  "count": 2,
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null]
      ]
    }
  ]
}
```

Запрос:

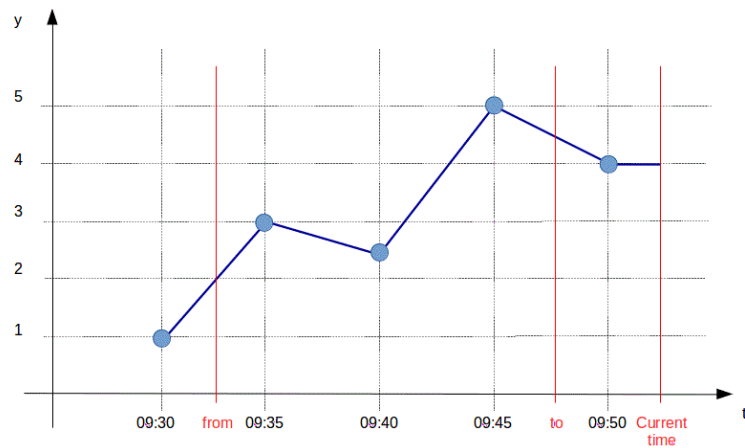
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "finish": "09:47:30",
  "count": 10,
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null]
      ]
    }
  ]
}
```

### **start, finish, maxCount, actual**

Возвращаются данные в количестве, заданном в `maxCount`. Если данных в базе меньше, чем задано в `maxCount`, возвращаются все данные из базы и флаг `excess` устанавливается в `false`. Если данных в базе больше, чем задано в `maxCount`, интервал времени разбивается на отрезки в соответствии с `maxCount`. Данные интерполируются/заполняются. Если на метку `start` нет данных, возвращается `null`. Если на метку `finish` нет данных, возвращается последнее имеющееся значение перед меткой времени `finish`.



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "finish": "09:47:30",
  "maxCount": 10,
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
      ]
    }
  ]
}
```

Запрос:

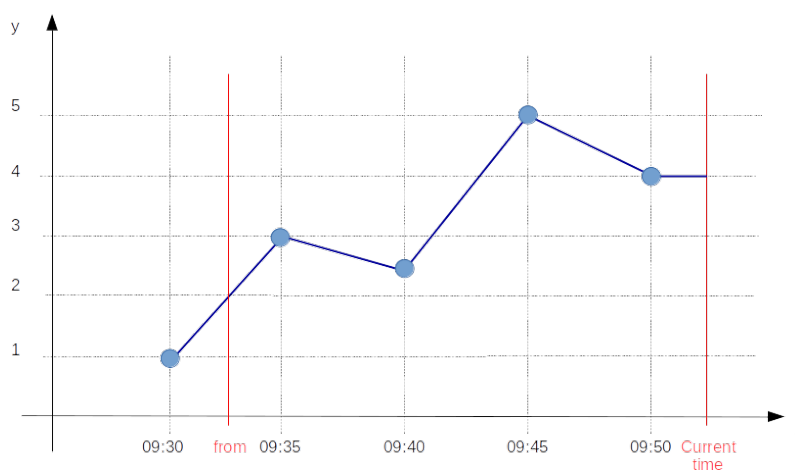
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "finish": "09:47:30",
  "maxCount": 2,
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "excess": true,
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [null, "09:32:30", null],
        [5, "09:47:30", null]
      ]
    }
  ]
}
```

## Ключ value

Ключ value используется для фильтрации запрошенных данных. Применяется в том случае, когда необходимо получить конкретные значения тэга.



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "value": [3]
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [3, "09:42:00", null]
      ]
    }
  ]
}
```

Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "value": [3, 4]
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [3, "09:41:00", null],
        [4, "09:43:00", null],
        [4, "09:50:00", null],
        [4, "09:53:00", null]
      ]
    }
  ]
}
```

## value и actual

В этом случае запрос при фильтрации данных будет учитывать только реальные данные, не интерполируя промежуточные значения.

Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "value": [3, 4],
  "actual": true
}
```

Ответ:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [3, "09:35:00", null],
        [4, "09:50:00", null]
      ]
    }
  ]
}
```

## null в истории данных

Важным моментом при работе с данными является случай, когда значением тэга является null.

Получение значения null в качестве данных тэга возможно, в первую очередь, тогда, когда тэг создан и в его историю ещё ничего не записано (для этого атрибут тэга `prsTagDefaultValue` должен быть пустым).

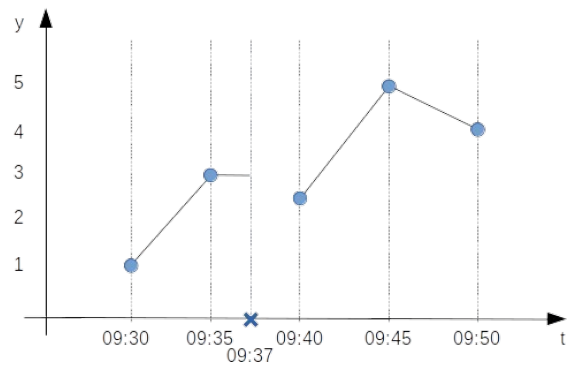
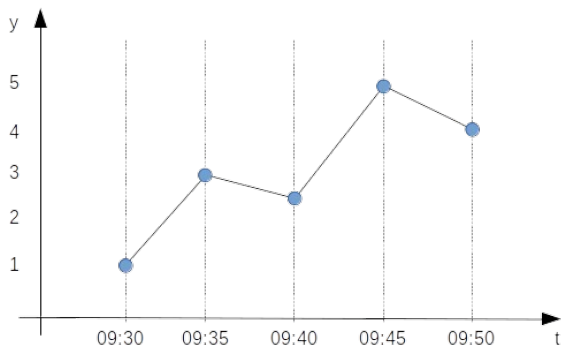
Кроме того, значение null записывается в тэг, когда:

1. Тэг привязан к источнику данных и с коннектором, записывающим данные в тэг, рвётся связь. В этом случае Платформа запишет в тэг значение null, код качества значения  $q = 100$ .
2. Платформа завершает работу. В этом случае во все тэги записывается значение null, код качества значения  $q = 101$ .
3. Тэг привязан к источнику данных и у коннектора рвётся связь с поставщиком данных. В этом случае коннектор запишет в тэг значение null с кодом качества значения  $q = 102$ .
4. Тэг привязан к источнику данных и коннектор получает от поставщика данных значение null.
5. В тэг было записано значение null командой `data/set`.

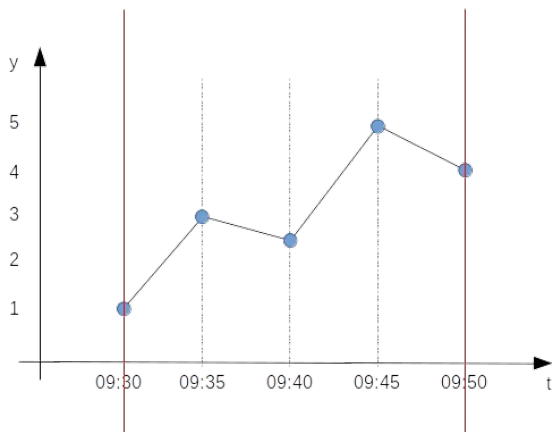
Считается, что с момента записи в тэг значения null до записи следующего значения мы ничего не знаем о данных тэга, в них как бы образовывается дыра. Поэтому значения тэга не интерполируются на этом промежутке.

При запросе данных за период null возвращается клиенту, если попадает в этот промежуток.

На картинке ниже показывается, как интерпретируются данные в случае, если в истории есть null.

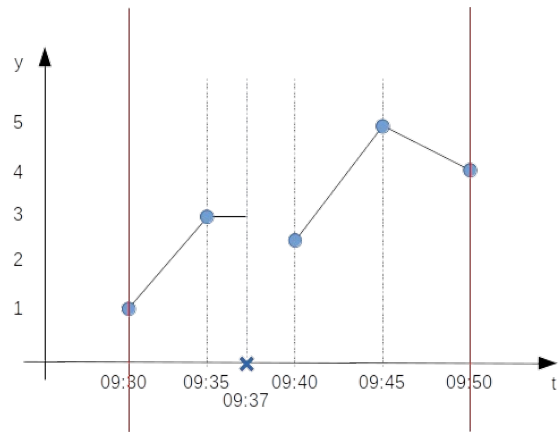


Далее - несколько примеров запросов к данным, показанным на картинке выше с ответами.



from

to



from

to

Запрос:

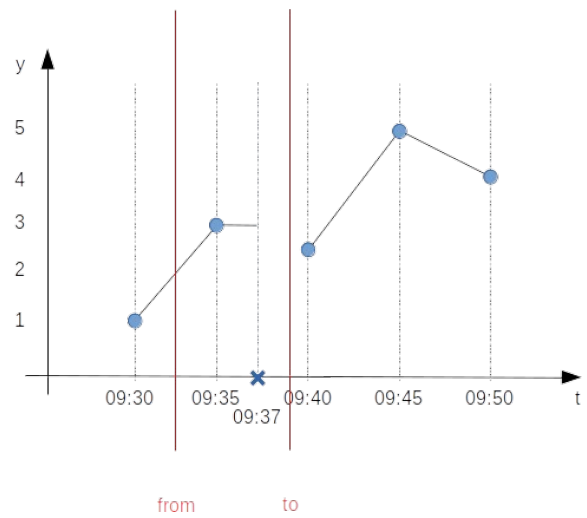
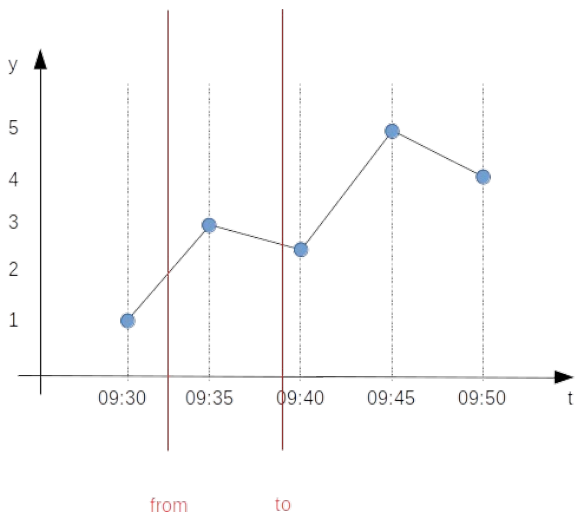
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:30:00",
  "finish": "09:50:00"
}
```

Ответ без ``null``:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [1, "09:30:00", null],
        [3, "09:35:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null]
      ]
    }
  ]
}
```

Ответ с ``null``:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [1, "09:30:00", null],
        [3, "09:35:00", null],
        [null, "09:37:00", null],
        [2.5, "09:40:00", null],
        [5, "09:45:00", null],
        [4, "09:50:00", null]
      ]
    }
  ]
}
```



Запрос:

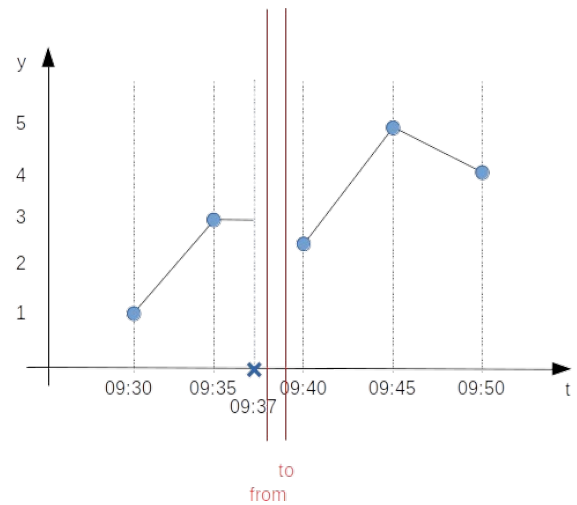
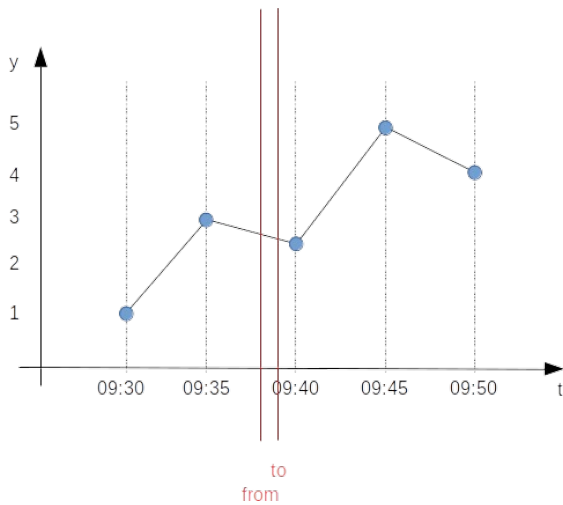
```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:30",
  "finish": "09:39:00"
}
```

Ответ без ``null``:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [2, "09:32:30", null],
        [3, "09:35:00", null],
        [2.6, "09:39:00", null]
      ]
    }
  ]
}
```

Ответ с ``null``:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [2, "09:32:30", null],
        [3, "09:35:00", null],
        [null, "09:37:00", null],
        [null, "09:39:00", null]
      ]
    }
  ]
}
```



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:38:00",
  "finish": "09:39:00"
}
```

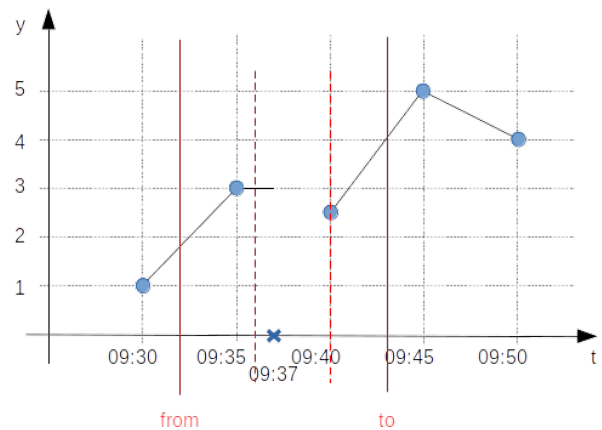
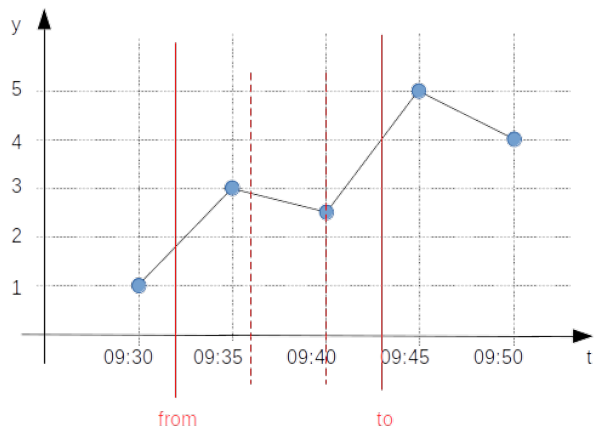
Ответ без ``null``:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [2.7, "09:38:00", null],
        [2.6, "09:39:00", null]
      ]
    }
  ]
}
```

Ответ с ``null``:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [null, "09:38:00", null],
        [null, "09:39:00", null]
      ]
    }
  ]
}
```

Сделаем запрос с флагом `timeStep` в 4 минуты:



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:00",
  "finish": "09:43:00",
  "timeStep": 240000000
}
```

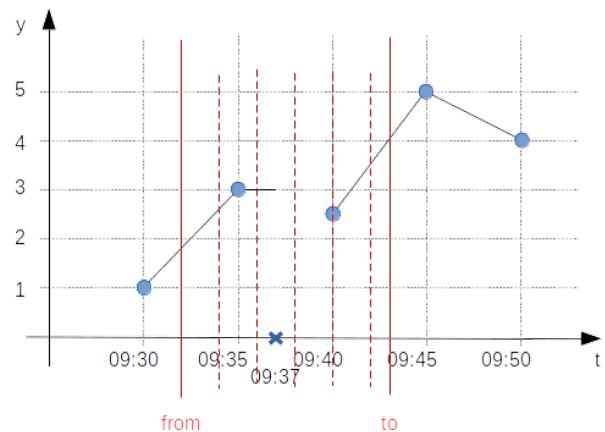
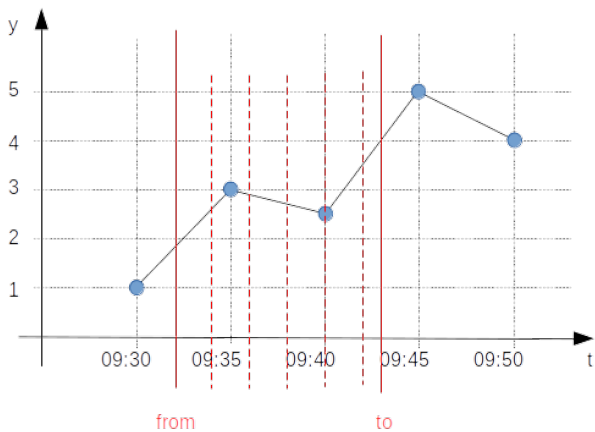
Ответ без ```null```:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [1.8, "09:32:00", null],
        [2.9, "09:36:00", null],
        [2.5, "09:40:00", null]
      ]
    }
  ]
}
```

Ответ с ```null```:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [1.8, "09:32:00", null],
        [3, "09:36:00", null],
        [2.5, "09:40:00", null]
      ]
    }
  ]
}
```

Теперь сделаем запрос с флагом timeStep в 2 минуты:



Запрос:

```
{
  "tagId": "1500c712-726a-103e-9264-a5021ec",
  "format": true,
  "start": "09:32:00",
  "finish": "09:43:00",
  "timeStep": 12000000
}
```

Ответ без ``null``:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [1.8, "09:32:00", null],
        [2.6, "09:34:00", null],
        [2.9, "09:36:00", null],
        [2.7, "09:38:00", null],
        [2.5, "09:40:00", null],
        [3.5, "09:42:00", null]
      ]
    }
  ]
}
```

Ответ с ``null``:

```
{
  "data": [
    {
      "tagId": "1500c712-726a-103e-9264-a5021ec2dae1",
      "data": [
        [1.8, "09:32:00", null],
        [2.6, "09:34:00", null],
        [3, "09:36:00", null],
        [null, "09:38:00", null],
        [2.5, "09:40:00", null],
        [3.5, "09:42:00", null]
      ]
    }
  ]
}
```

## Примеры работы с МПК-Пересвет

Рассмотрим самый простой пример.

### Простой пример с одним объектом

Для демонстрации работы с объектами, тегами, методами и экранами Grafana сделаем следующий пример.

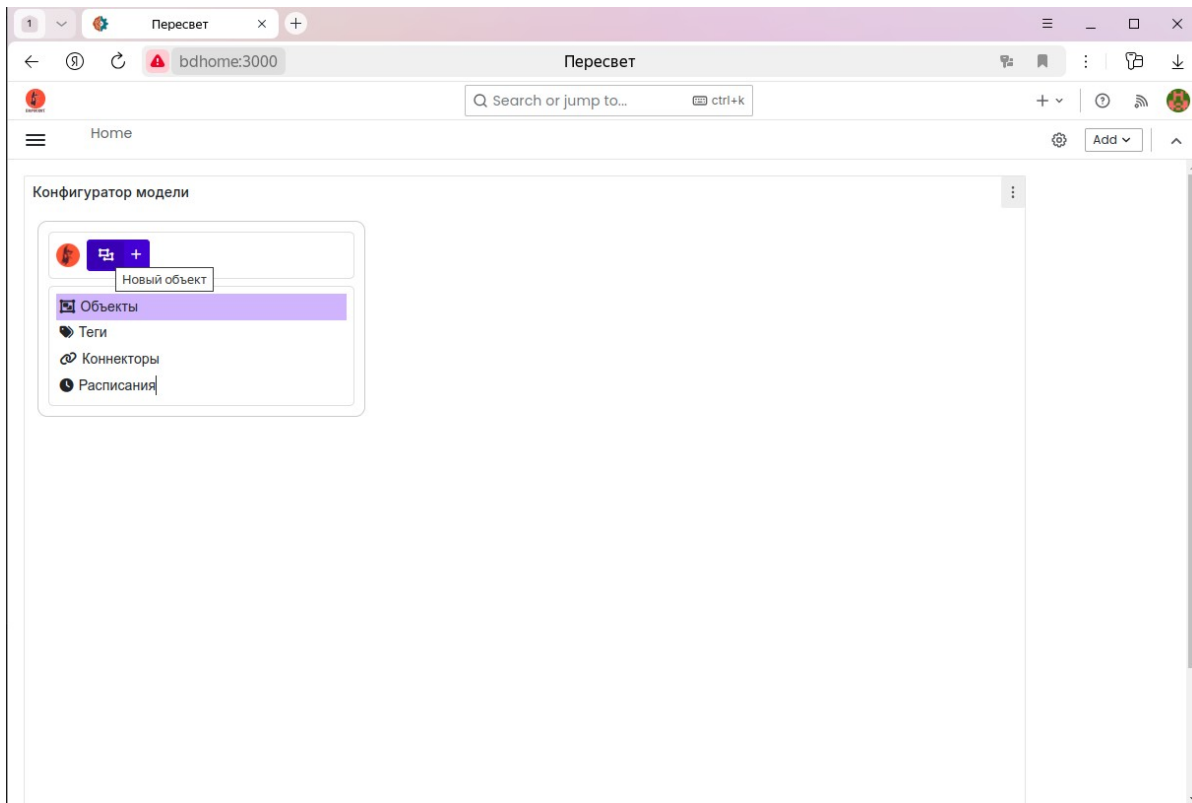
Допустим, у нас есть лампочка и мы можем тем или иным способом получить данные по току и напряжению (работу с коннекторами рассмотрим в следующих примерах).

Кроме тока и напряжения у лампочки есть третий, расчётный тег - мощность.

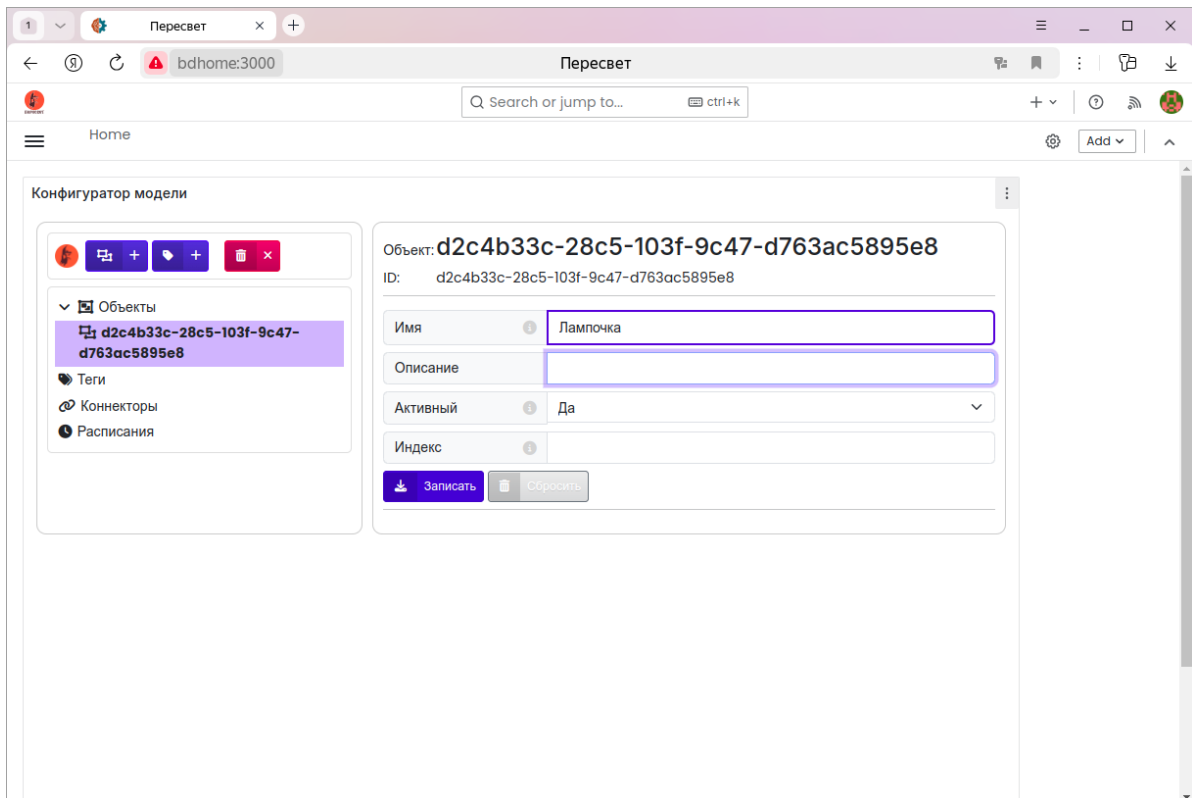
При изменении силы тока или напряжения мощность должна перерасчитываться.

### Объект

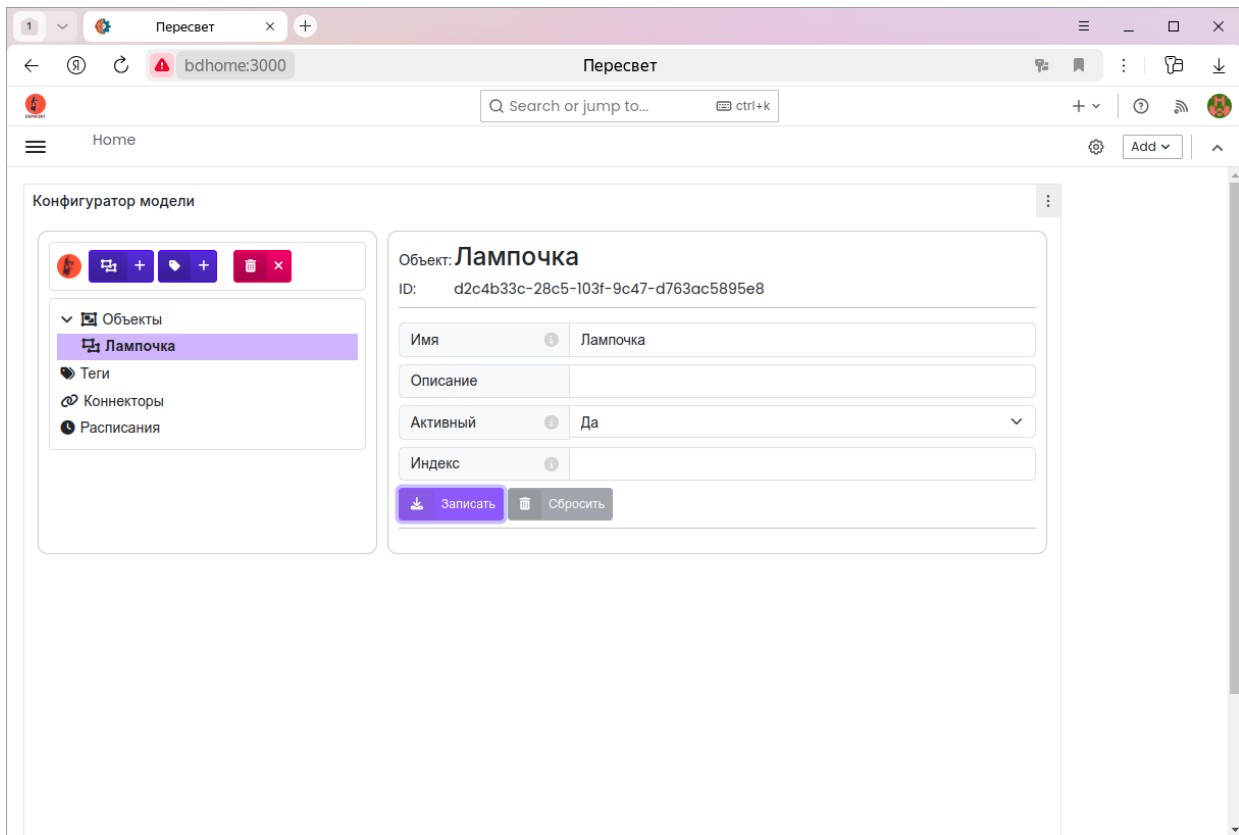
Открываем конфигуратор, выбираем папку «Объекты» и нажимаем кнопку «Новый объект»:



В появившемся окне свойств меняем имя объекта на «Лампочка»...

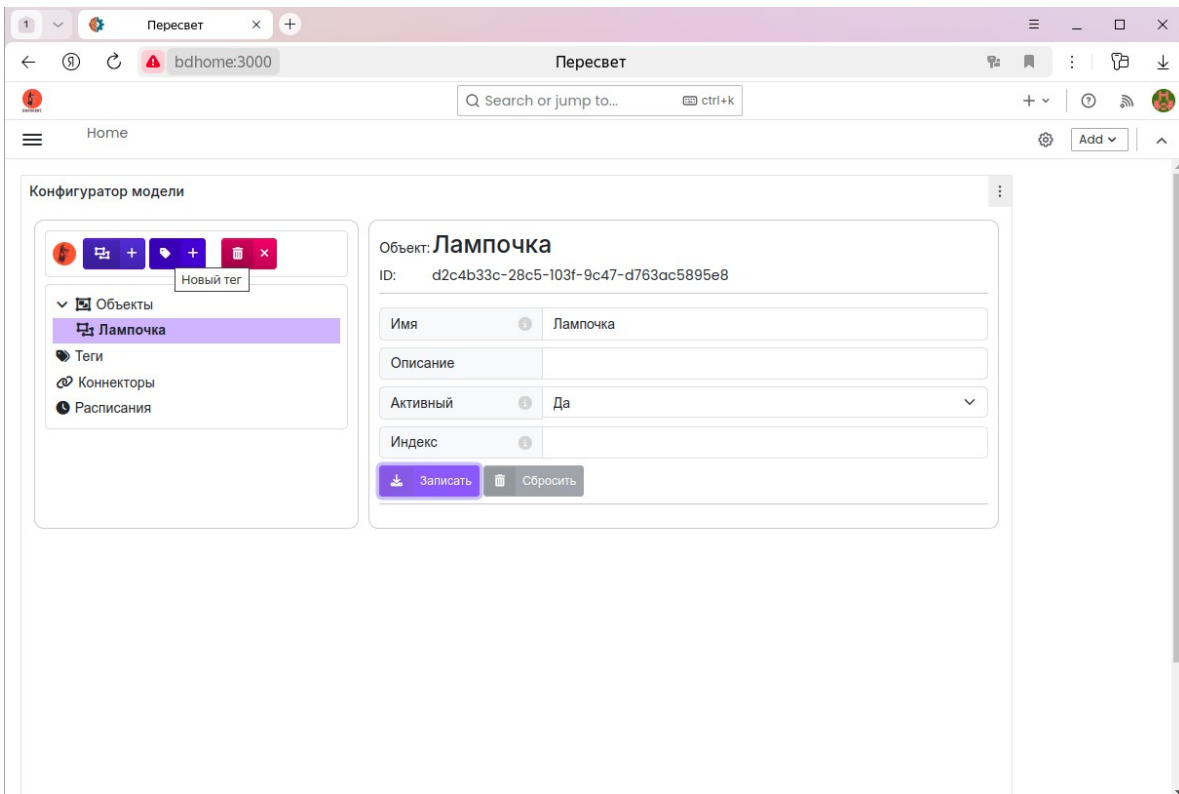


...и, нажав кнопку «Записать», сохраняем изменения:

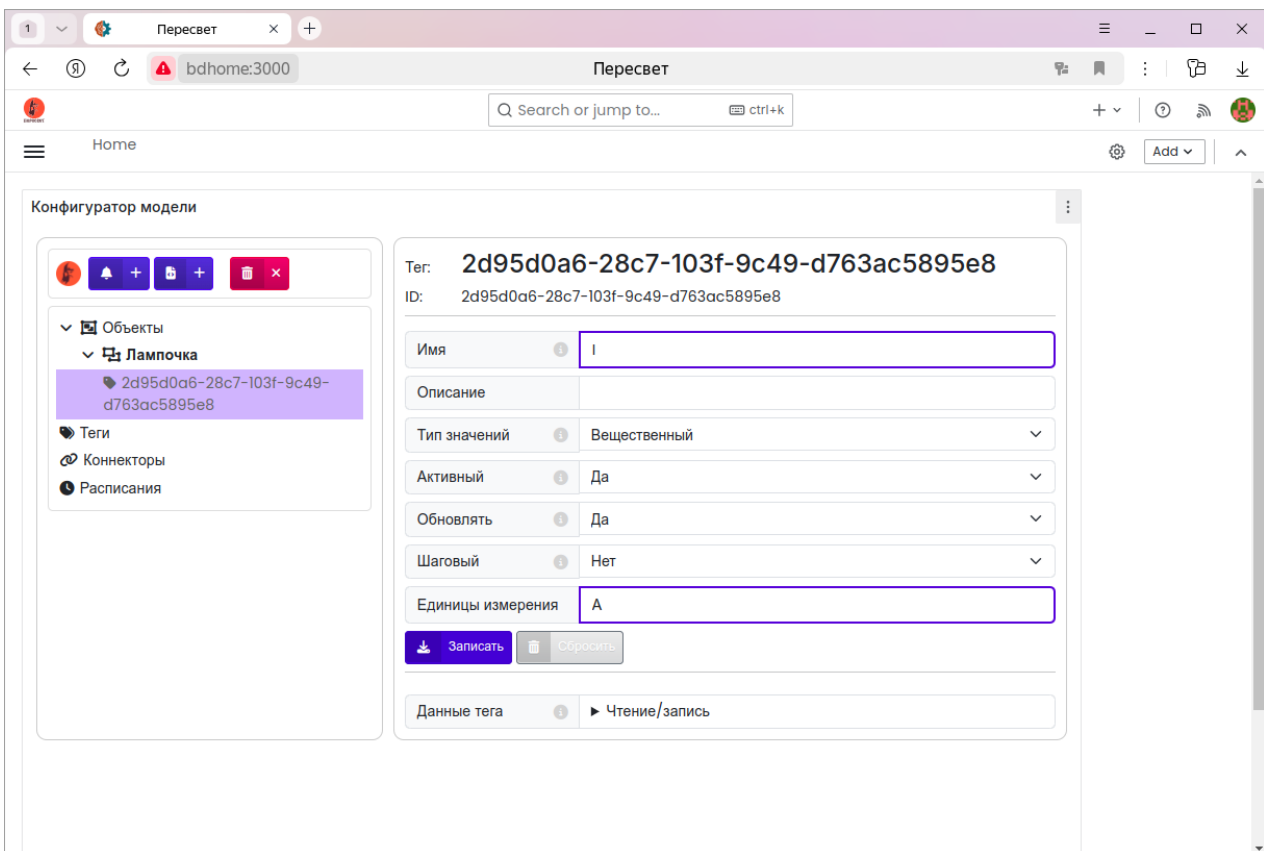


## Теги

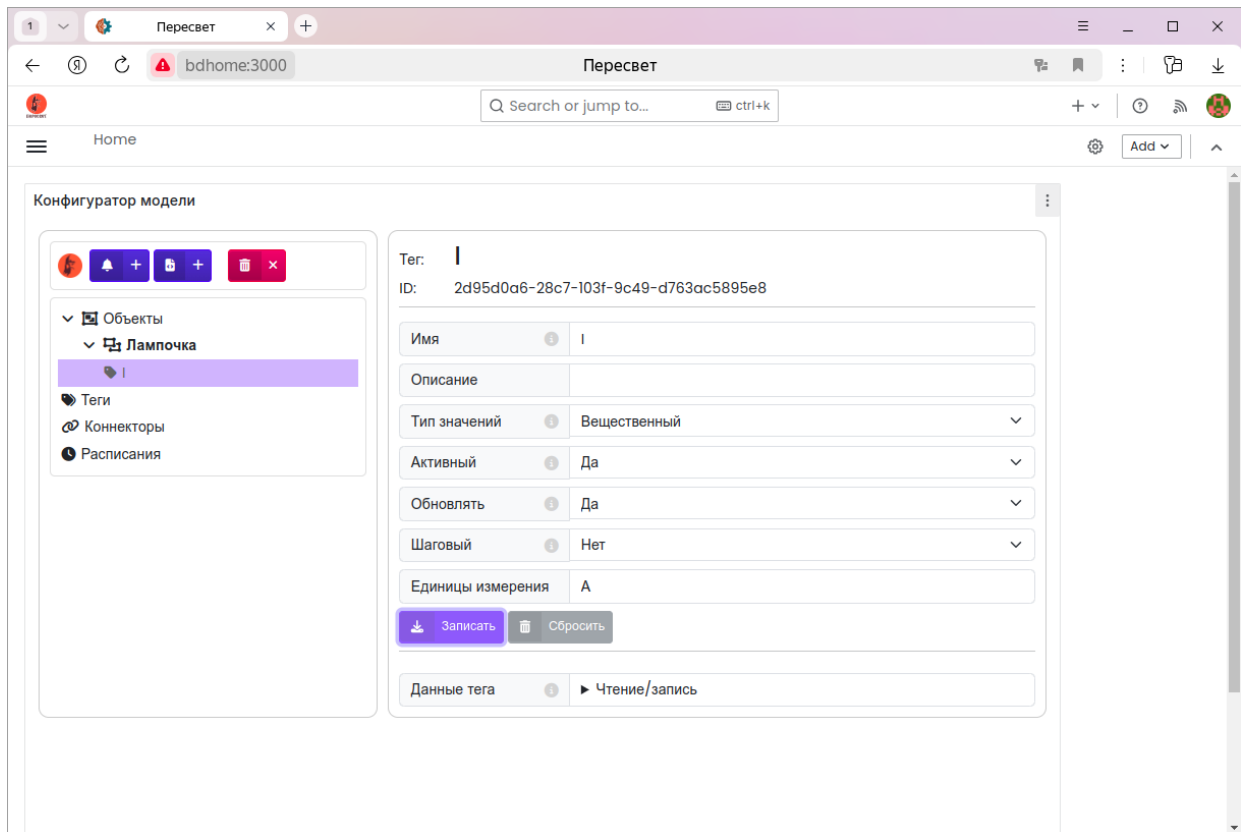
Оставляя текущим объектом «Лампочку», нажимаем кнопку «Новый тег»:



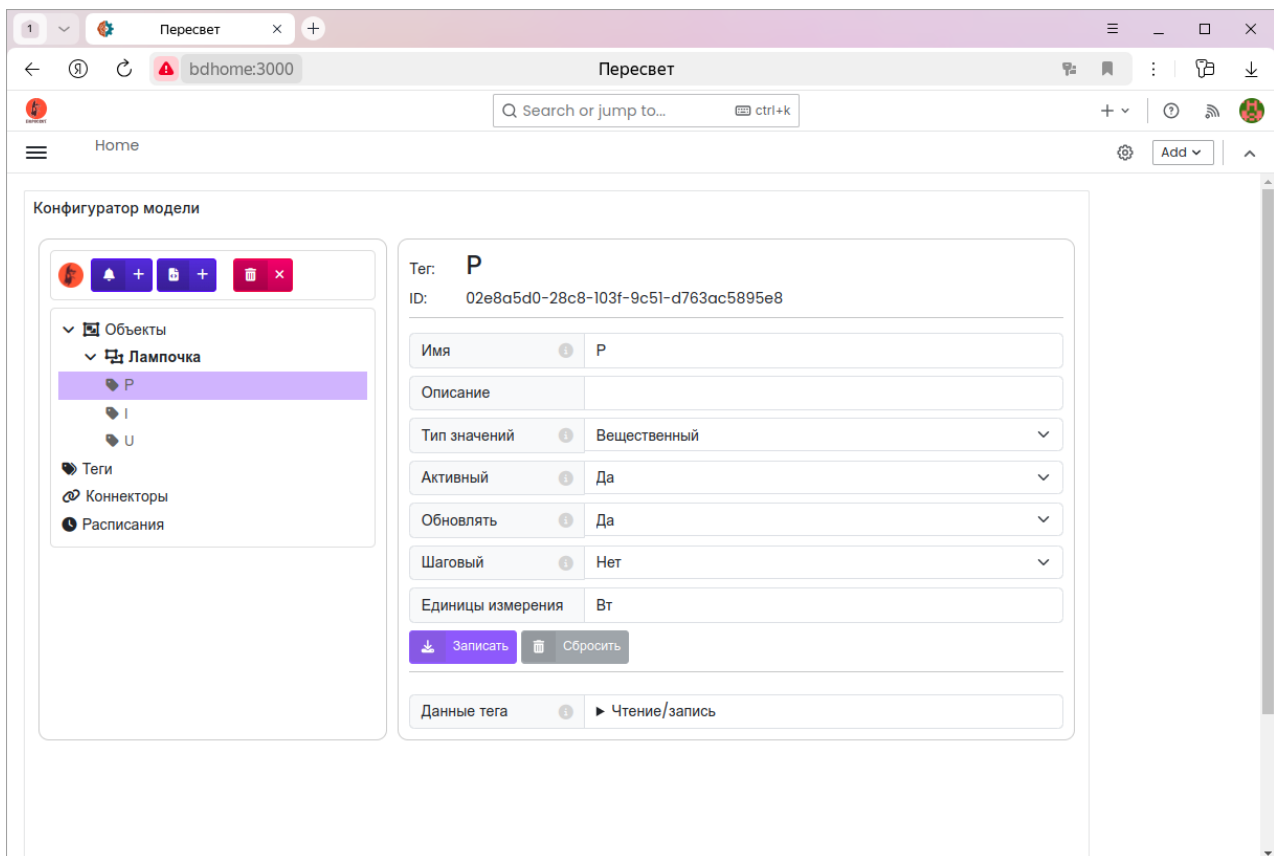
Меняем имя тега и единицы измерения...



...и, нажав кнопку «Записать», сохраняем изменения:



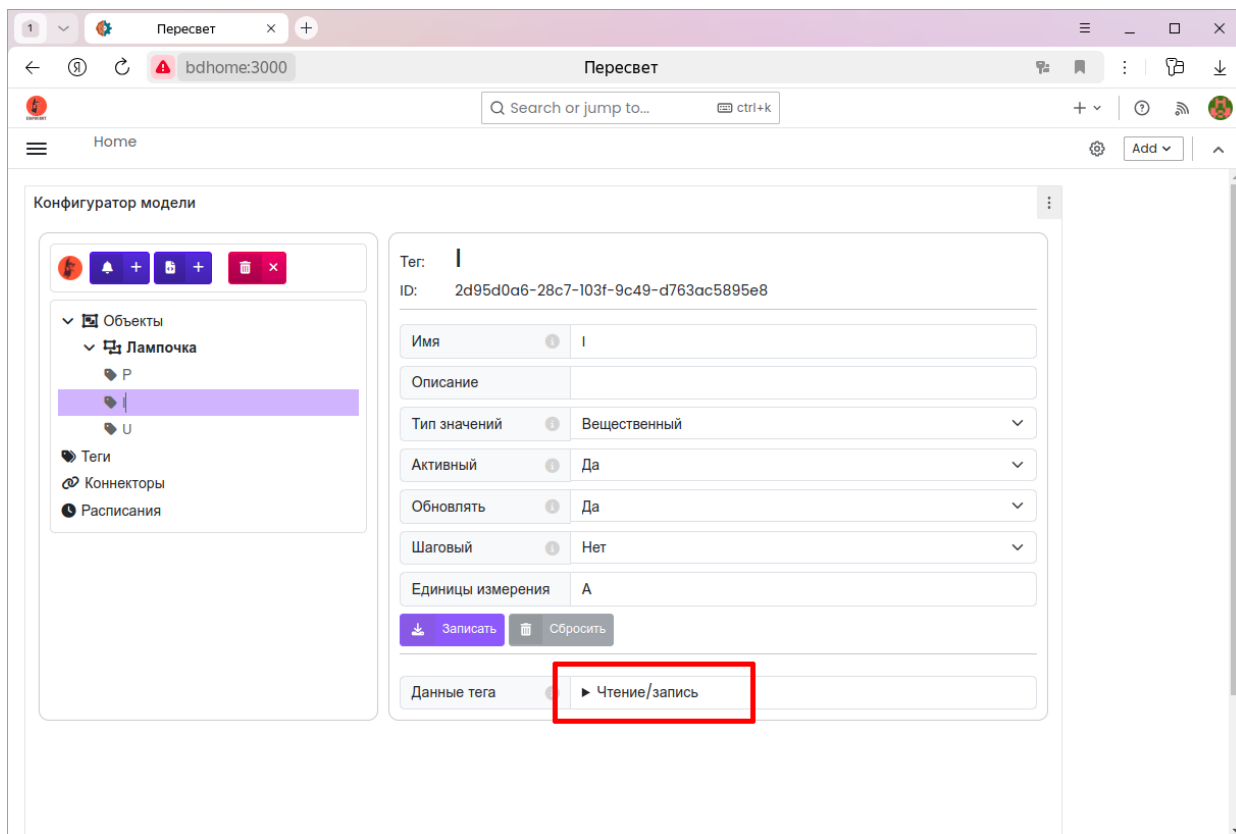
Аналогичным образом создаём ещё два тега, U и P:



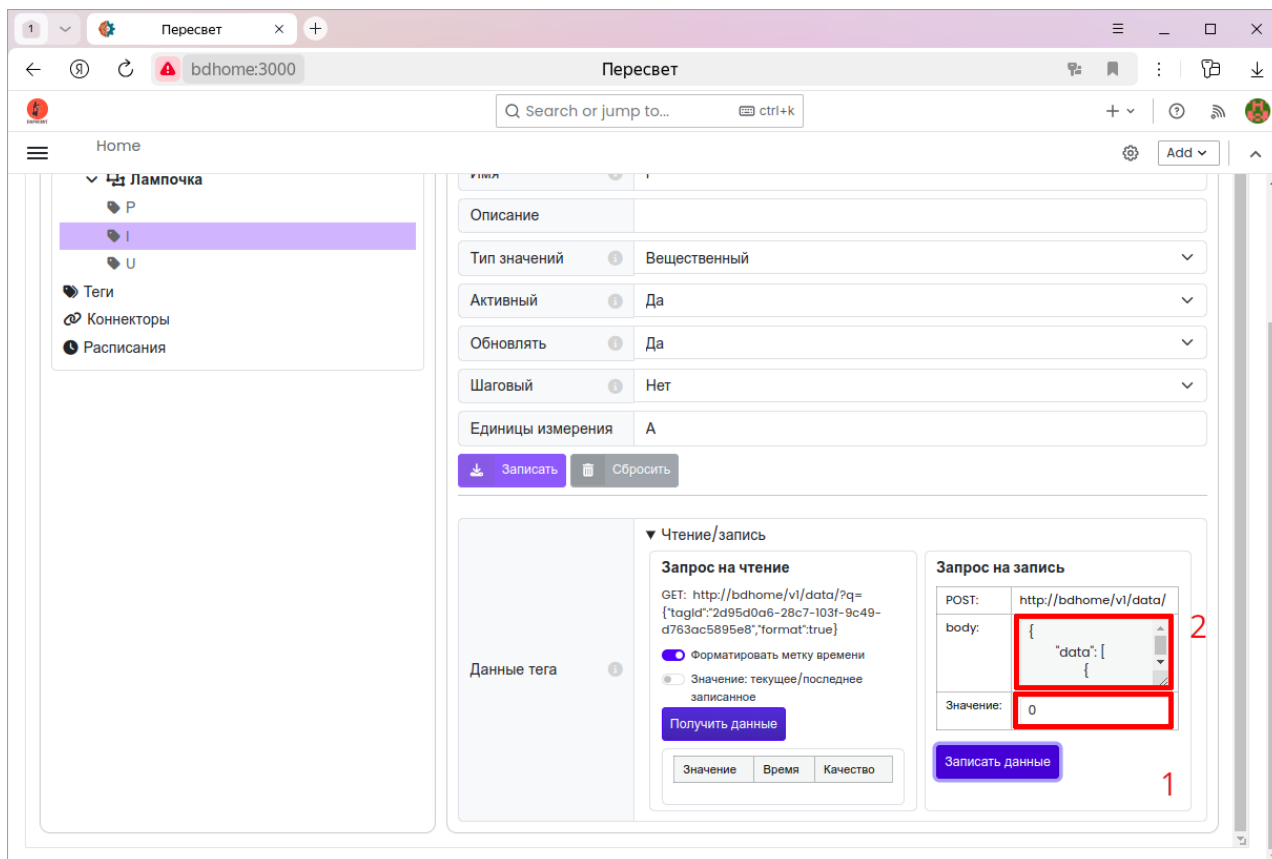
## Запись/чтение данных

Проверим, что данные в теги записываются и читаются.

Выберем тег «I» и откроем панель «Данные тега»:



1. В поле «Значение» впишем 0.
2. В поле «body» можно посмотреть сформированное тело запроса на запись.



Нажимаем кнопку «Записать», затем «Получить данные».

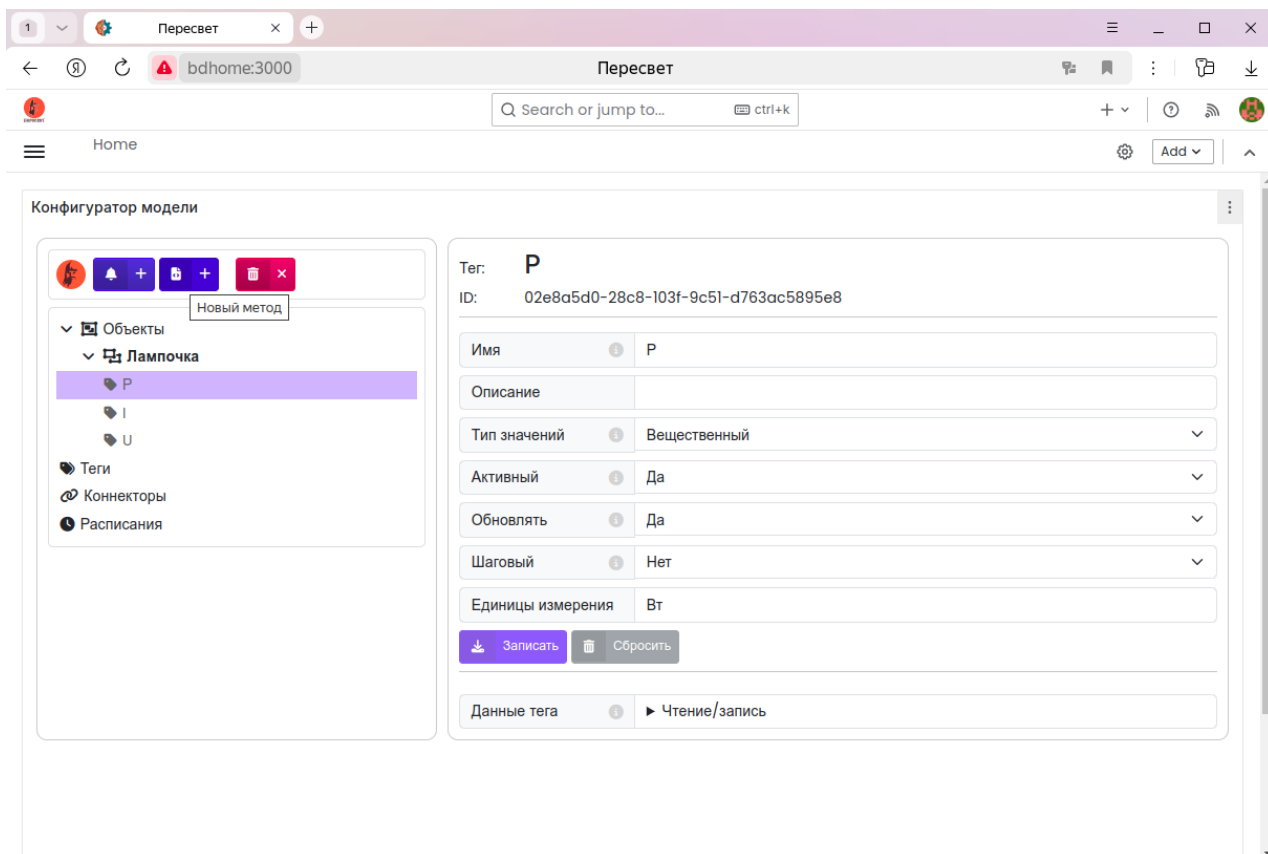
В таблице «Значение-время-качество» появится текущее значение тега «I»:

Screenshot of the 'Пересвет' web interface showing configuration for tag 'P'. The interface includes a sidebar with 'Теги', 'Коннекторы', and 'Расписания'. The main area shows configuration fields for 'Описание', 'Тип значений', 'Активный', 'Обновлять', 'Шаговый', and 'Единицы измерения'. Below these are 'Записать' and 'Сбросить' buttons. A 'Чтение/запись' section contains 'Запрос на чтение' and 'Запрос на запись' panels. The 'Запрос на чтение' panel shows a GET request and a table with one row of data: '0', '2024-10-27T19:14:04.184413+03:00', and 'null'. The 'Запрос на запись' panel shows a POST request and a 'Записать данные' button.

## Расчётный метод

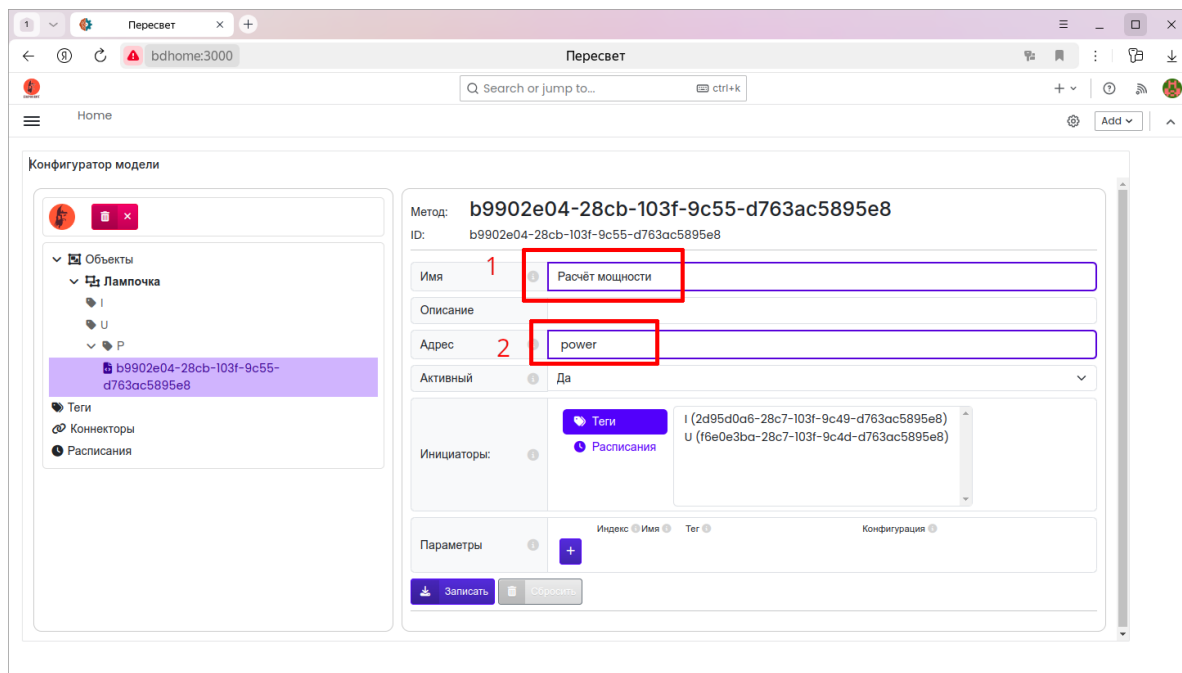
Создадим метод для расчёта мощности лампочки.

Для этого выберем тег «P» и нажмём кнопку «Новый метод»:



Затем:

1. Изменим имя метода
2. В поле «Адрес» введём название метода, под которым он будет зарегистрирован в системе для вызовов. Запомним это имя - **power**:



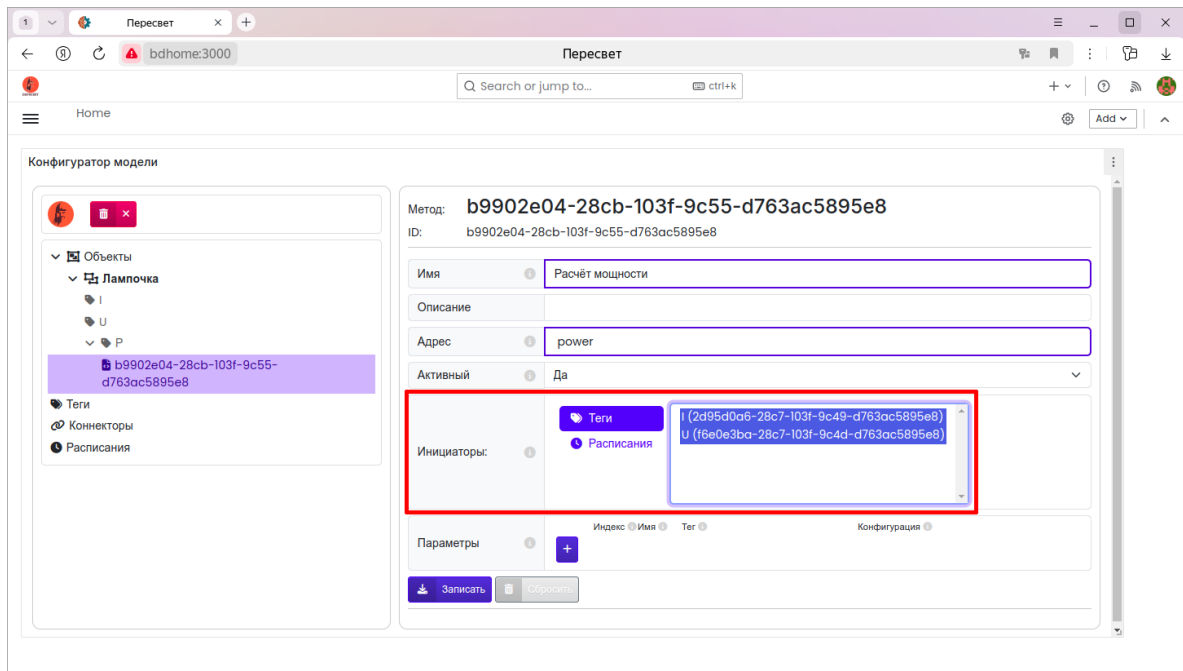
3. Блок «Инициаторы». Очень важный блок, в котором мы выбираем, какие события могут инициировать выполнение расчётного метода.

Здесь выберем оба предложенных тега: I и U, то есть изменения обоих этих тегов будут приводить к перерасчёту мощности.

---

*Обратите внимание, что в списке возможных инициаторов расчёта нет самого тега «P»: изменение тега не может приводить к перерасчёту его самого, так как это вызовет бесконечный цикл расчётов.*

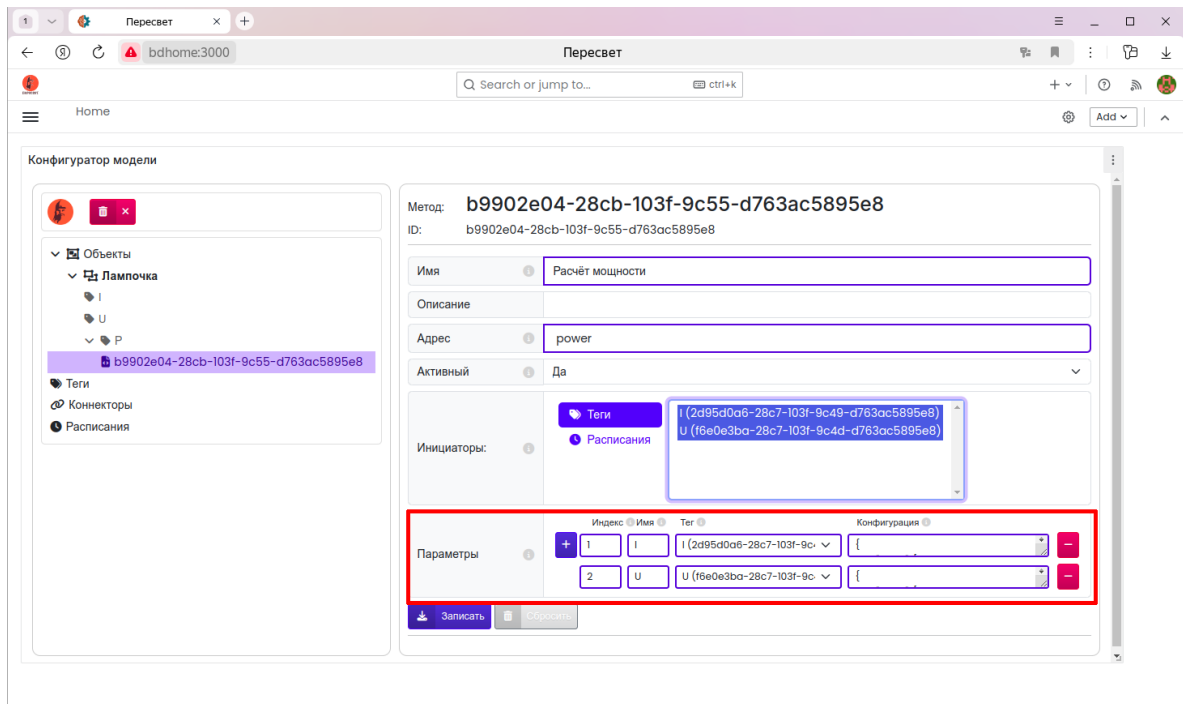
---



4. Укажем, какие данные необходимо передать методу для расчёта.

Данные, передаваемые в метод в каждом параметре - это результат запроса [получения данных](#).

Для расчёта мощности нужны текущие значения двух тегов: силы тока и напряжения. Поэтому создаём два параметра, дважды нажав на синюю кнопку с плюсом. Заполняем данные параметров как показано на картинке:



Здесь мы указали, что в метод передаются два параметра: первым (индекс = 1) - сила тока.

Вторым (индекс = 2) - напряжение.

Поле «Конфигурация» заполняется автоматически после выбора тега в поле «Тег».

#### Внимание

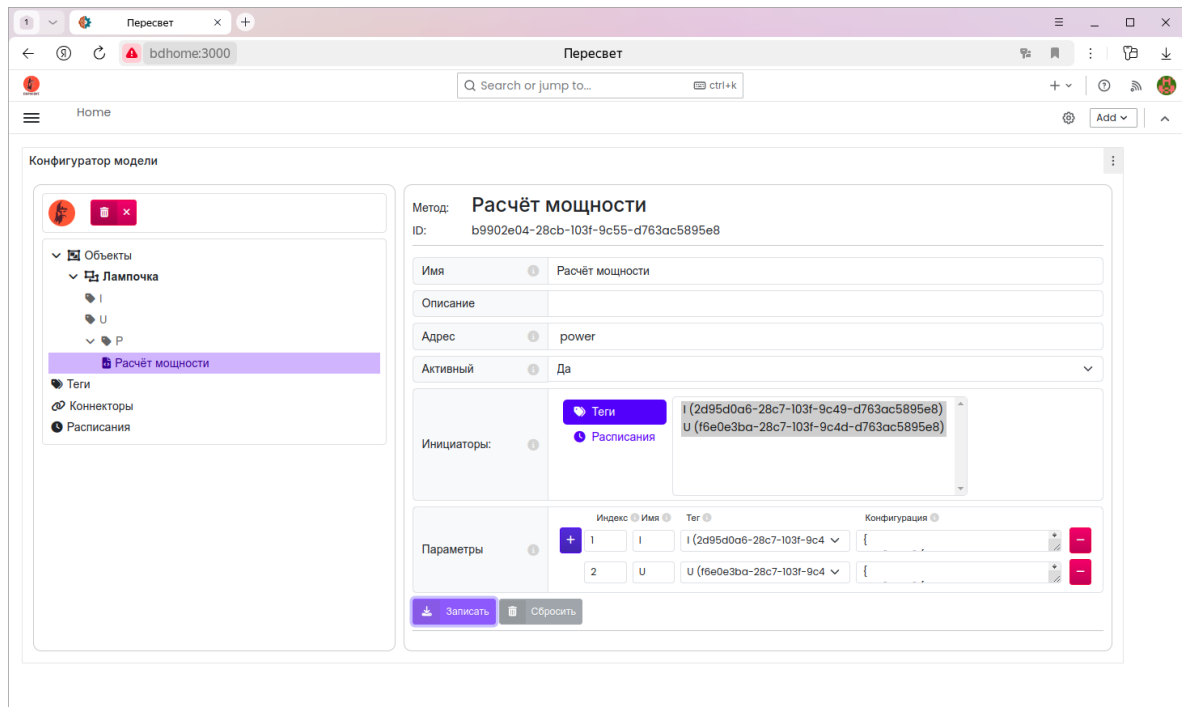
После изучения документации на запрос [получения данных](#) можно вручную править поле «Конфигурация», составляя сколь угодно сложные запросы получения данных. Поле «Конфигурация» можно расширять для удобства ввода текста.

Индекс	Имя	Тег	Конфигурация
1	I	I (2d95d0a6-28c7-103f-9c49-d763ac5895e8)	{ "tagId": [ "2d95d0a6-28c7-103f-9c49-d763ac5895e8" ] }
2	U	U (f6e0e3ba-28c7-103f-9c4d-d763ac5895e8)	{

Записать | Сбросить

*Очень важный момент, относящийся к расчётам значений тегов: тег мощности рассчитывается, когда изменяется значение тегов «I» и «U». Допустим, расчёт инициируется изменением тега «I» и метка времени нового значения «I» - «2024-10-27 10:00:00+03:00». В этом случае запрос данных в параметрах будет производиться также на метку времени «2024-10-27 10:00:00+03:00» и новое рассчитанное значение тега мощности также будет записано с этой меткой времени, не смотря на то, что сам расчёт, возможно, будет производиться в другое время.*

5. Нажимаем кнопку «Записать» для сохранения изменений:



Теперь напишем код метода.

В папке `methods` уже написан метод для этого примера.

Откроем файл `methods/test_method.py` и рассмотрим метод `calc_power`:

```
@rpc("power")
async def calc_power(I: dict, U: dict) -> float:
    """Метод возвращает произведение двух параметров.
    """
    print(f"I: {json.dumps(I, indent=4)}")
    print(f"U: {json.dumps(U, indent=4)}")

    cur_I = I["data"][0]["data"][0][0]
    cur_U = U["data"][0]["data"][0][0]

    if cur_I == None or cur_U == None:
        return 0

    return cur_I * cur_U
```

*Первая строка: `@rpc("power")`: именно «power» мы вводили в поле «Адрес» при создании метода!*

Далее - строка объявления метода: `async def calc_power(I: dict, U: dict) -> float:`. Первым параметром указана сила тока, вторым - напряжение. Точно так же, как мы указывали индексы у параметров.

Далее метод просто для удобства выводит в консоль пришедшие данные, затем рассчитывает мощность как произведение двух параметров.

## Запуск расчётного метода

Для запуска метода необходимо установить пакеты, указанные в файле `methods/requirements.txt` либо для всех пользователей:

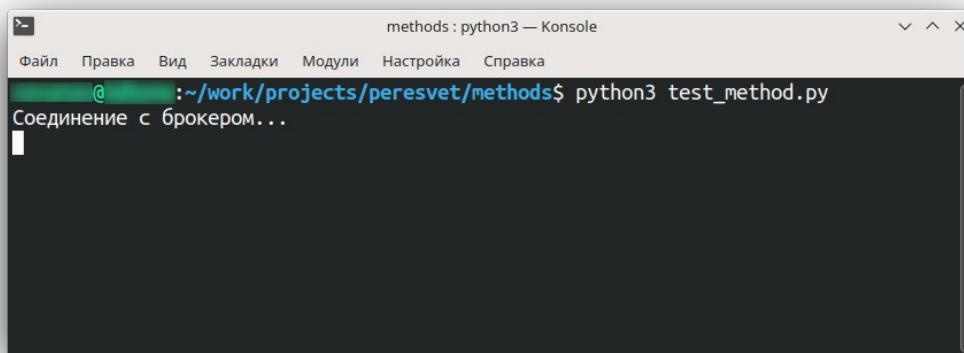
```
$ pip3 install -r requirements.txt
```

Либо создав виртуальную среду проекта.

После установки пакетов запускаем в терминале скрипт в папке `methods`:

```
$ python3 test_method.py
```

Результат должен быть примерно таким:



The screenshot shows a terminal window titled "methods : python3 — Konsole". The terminal content is as follows:

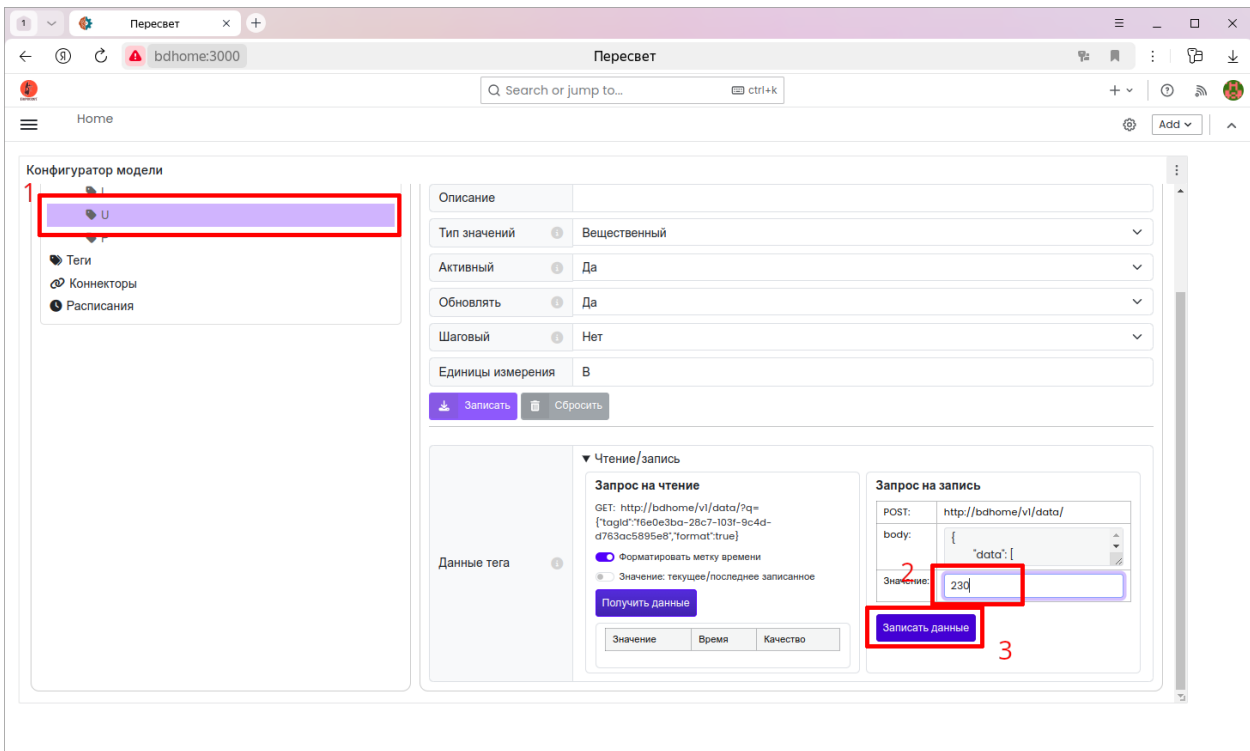
```
methods : python3 — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
@ ~/work/projects/peresvet/methods$ python3 test_method.py
Соединение с брокером...
```

## Проверим расчёт

Помним, что расчёт мощности инициируется изменениями значений тока и напряжения.

Запишем в тег «U» значение 230.

Для этого выберем в иерархии тег «U» и запишем в него требуемое значение:



В консоли, в которой мы запустили метод, появится вывод:

```

methods : python3 — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
@ ~/work/projects/peresvet/methods$ python3 test_method.py
Соединение с брокером...
I: {
  "data": [
    {
      "tagId": "2d95d0a6-28c7-103f-9c49-d763ac5895e8",
      "data": [
        0.0,
        1730062407441520,
        null
      ]
    }
  ]
}
U: {
  "data": [
    {
      "tagId": "f6e0e3ba-28c7-103f-9c4d-d763ac5895e8",
      "data": [
        230.0,
        1730062407441520,
        null
      ]
    }
  ]
}
  
```

Проверим значение тега мощности «P». Оно должно быть равно 0, так как перед этим в тег «I» мы записали 0:

Конфигуратор модели

- U
- P**
  - Расчёт мощности
- Теги
- Коннекторы
- Расписания

Тип значений: Вещественный

Активный: Да

Обновлять: Да

Шаговый: Нет

Единицы измерения: Вт

Записать Сбросить

Чтение/запись

Запрос на чтение

GET: `http://bdhome/v1/data/?q=[*tagId*:02e8a5d0-28c8-103f-9c51-d763ac5895e8,*format:true]`

Форматировать метку времени

Значение: текущее/последнее записанное

Получить данные

Значение	Время	Качество
0	2024-10-27 12:53:52.227721+03:00	null

Запрос на запись

POST: `http://bdhome/v1/data/`

body: `{ "data": [ ] }`

Значение:

Записать данные

Теперь запишем в тег «I» значение 1:

Конфигуратор модели

- I**
- U
- P
  - Расчёт мощности
- Теги
- Коннекторы
- Расписания

Описание

Тип значений: Вещественный

Активный: Да

Обновлять: Да

Шаговый: Нет

Единицы измерения: А

Записать Сбросить

Чтение/запись

Запрос на чтение

GET: `http://bdhome/v1/data/?q=[*tagId*:2d95d0a6-28c7-103f-9c49-d763ac5895e8,*format:true]`

Форматировать метку времени

Значение: текущее/последнее записанное

Получить данные

Значение	Время	Качество
1	2024-10-28 10:01:16.386218+03:00	null

Запрос на запись

POST: `http://bdhome/v1/data/`

body: `{ "data": [ ] }`

Значение:

Записать данные

И проверим значение мощности, которое должно быть равно 230 Вт:

Конфигуратор модели

- I
- U
- P
- Расчёт мощности
- Теги
- Коннекторы
- Расписания

Описание

Тип значений: Вещественный

Активный: Да

Обновлять: Да

Шаговый: Нет

Единицы измерения: Вт

Записать Сбросить

Чтение/запись

Запрос на чтение

GET: `http://bdhome/v1/data/?q=[\"tagId\":\"02e8a5d0-28c8-103f-9c51-d763ac5895e8\", \"format\":true]`

Форматировать метку времени

Значение: текущее/последнее записанное

Получить данные

Значение	Время	Качество
230	2024-10-28T00:03:00.294282+03:00	null

Запрос на запись

POST: `http://bdhome/v1/data/`

body: `{ \"data\": [ ] }`

Значение:

Записать данные

## Отображение данных в Grafana

Данные из платформы можно получать запросами, описанными в разделе «Исторические данные».

Данные тегов (как, впрочем, и всё общение с платформой) приходят в виде json и для манипулирования этими данными в Grafana рекомендуется изучить языки манипулирования json-данными [JSONata](#) или [JSONPath](#).

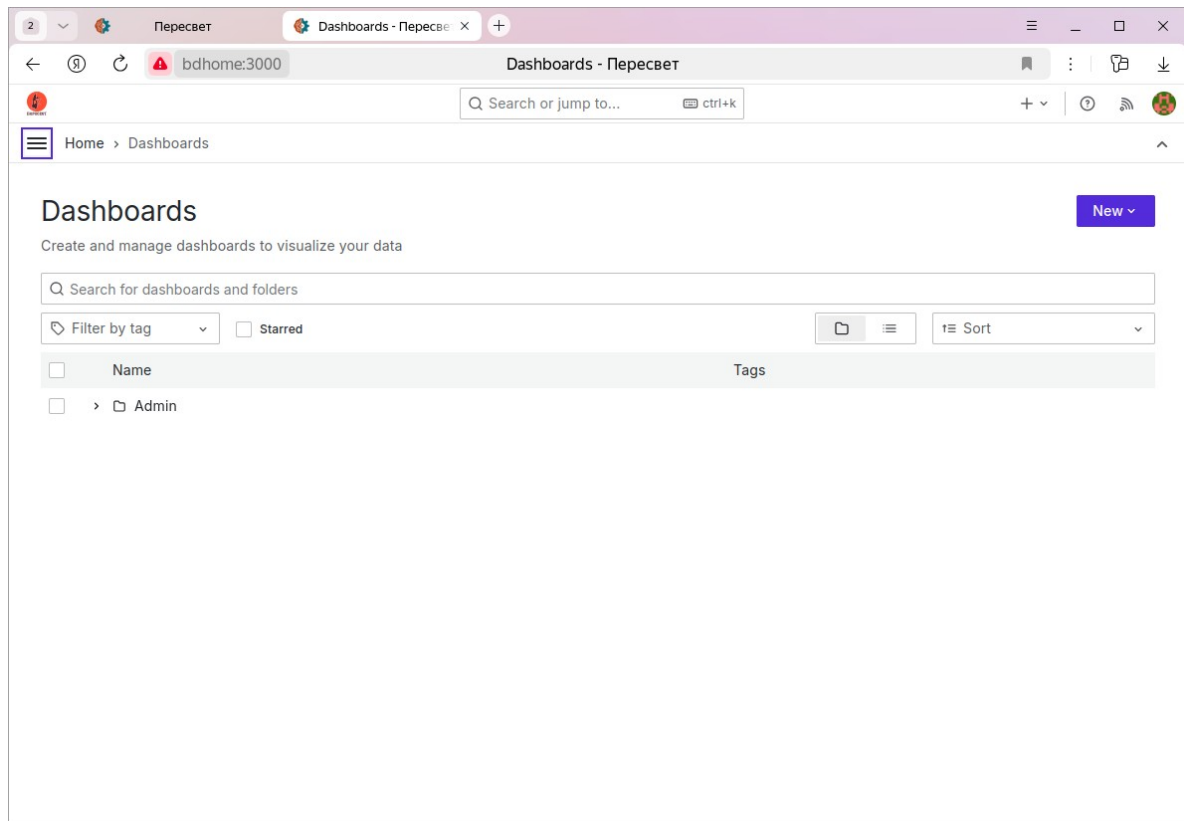
Предпочтительнее JSONata, так как обладает большими возможностями.

Настроим отображение:

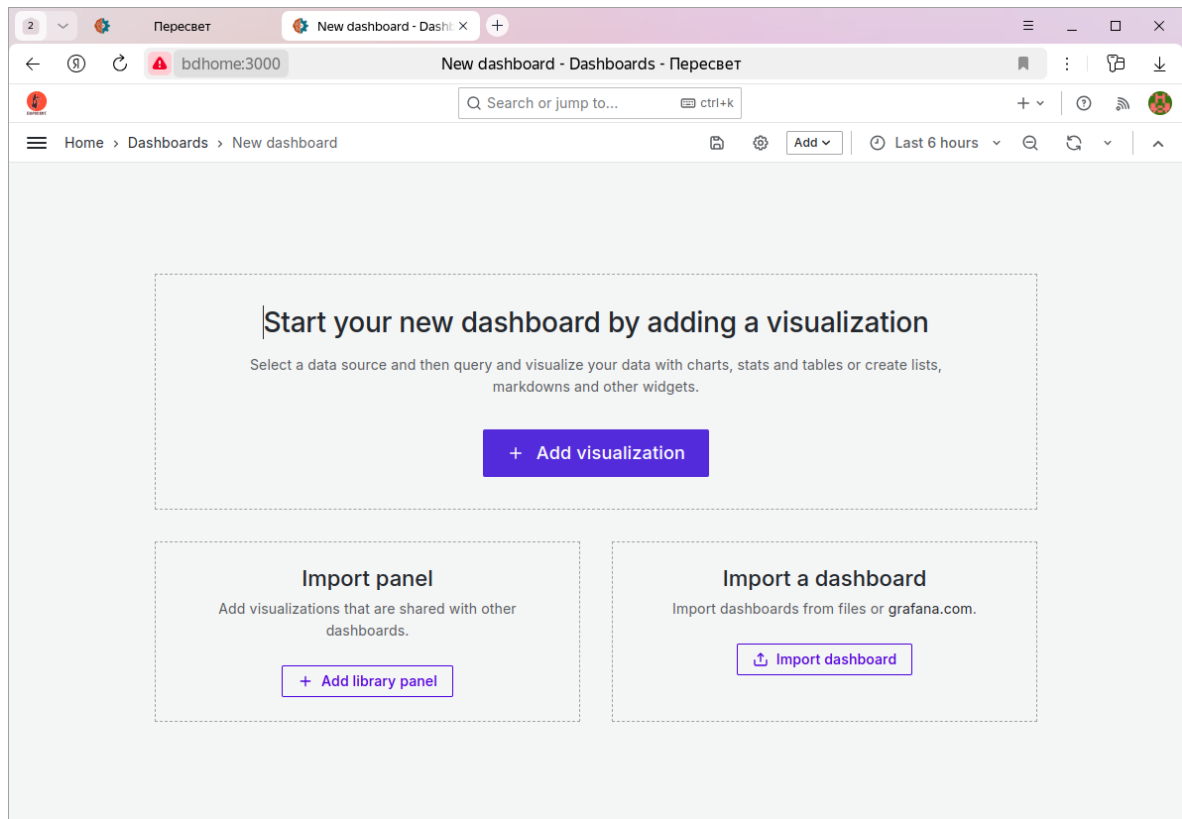
- текущих значений тегов «I», «U» и «P»;
- трендов значений этих тегов;

1. Создадим новую доску данных.

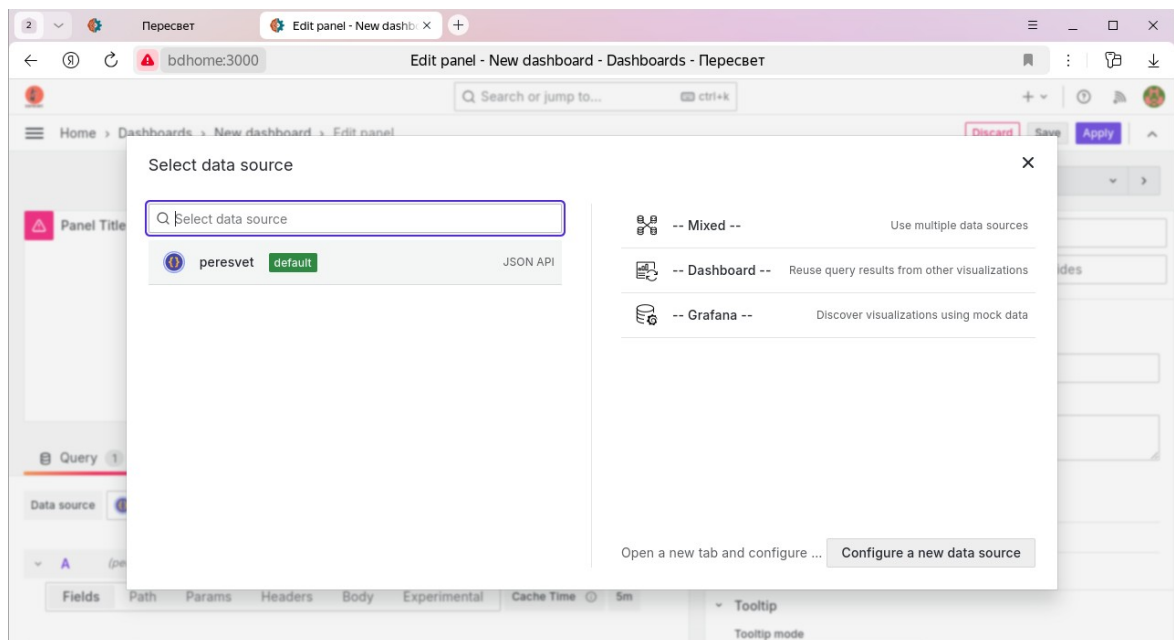
Для этого заходим в меню «Home → Dashboards» и нажимаем кнопку «New»



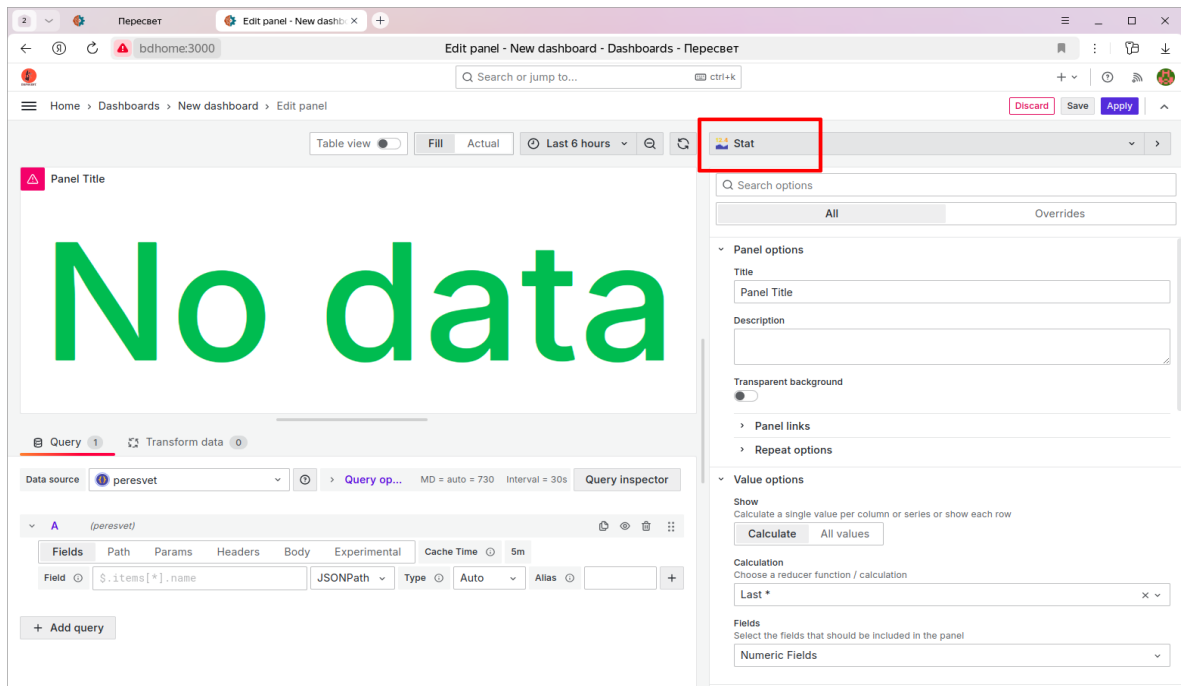
Выбрав в выпадающем меню команду «New dashboard», попадаем на экран создания новой панели:



...и выбираем источник данных «peresvet»:

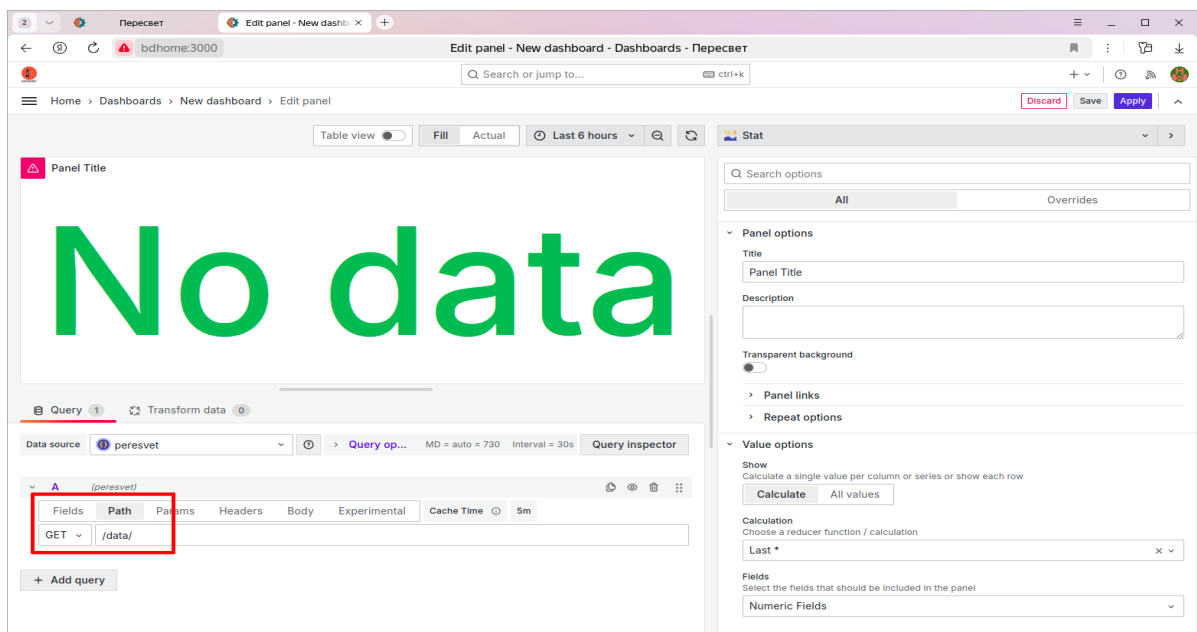


2. В появившемся окне настройки панели выберем тип панели «Stat».



3. Настроим путь для получения данных, вписав `/data/` на закладке Path.

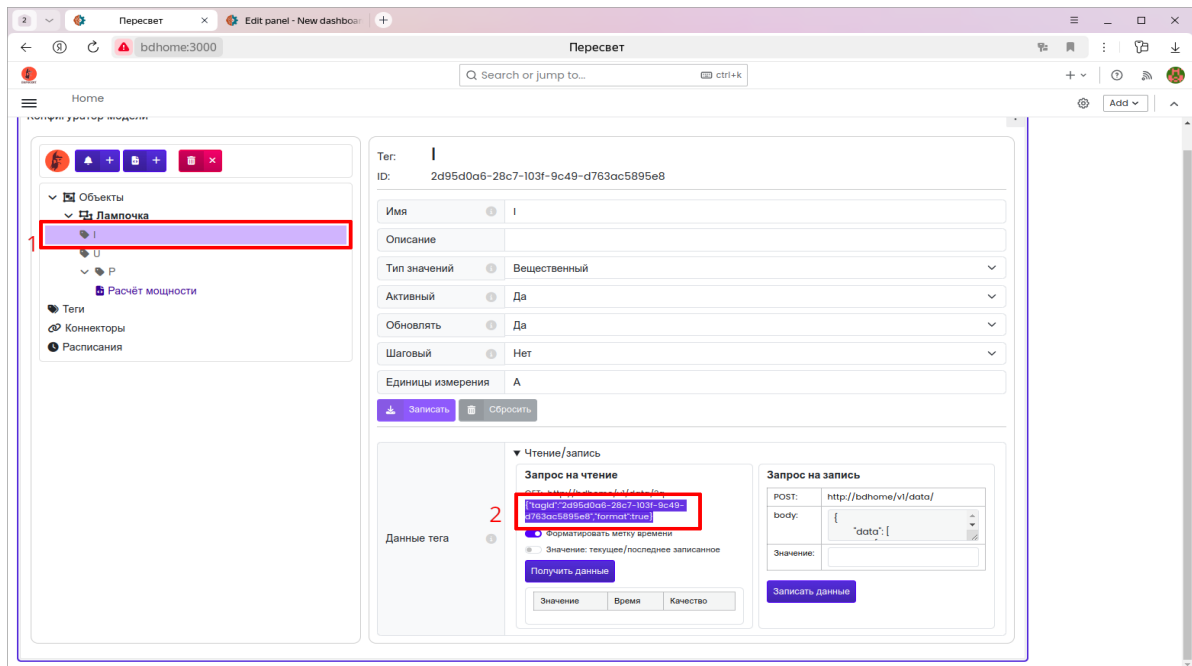
*По умолчанию в источнике данных «peresvet» настроен путь к платформе: `http://<server>/v1/`. Добавляя к этому пути расширения `data`, `objects`, `tags` и т.д., мы получаем возможность работать не только с данными, но и вообще со всеми сущностями платформы.*



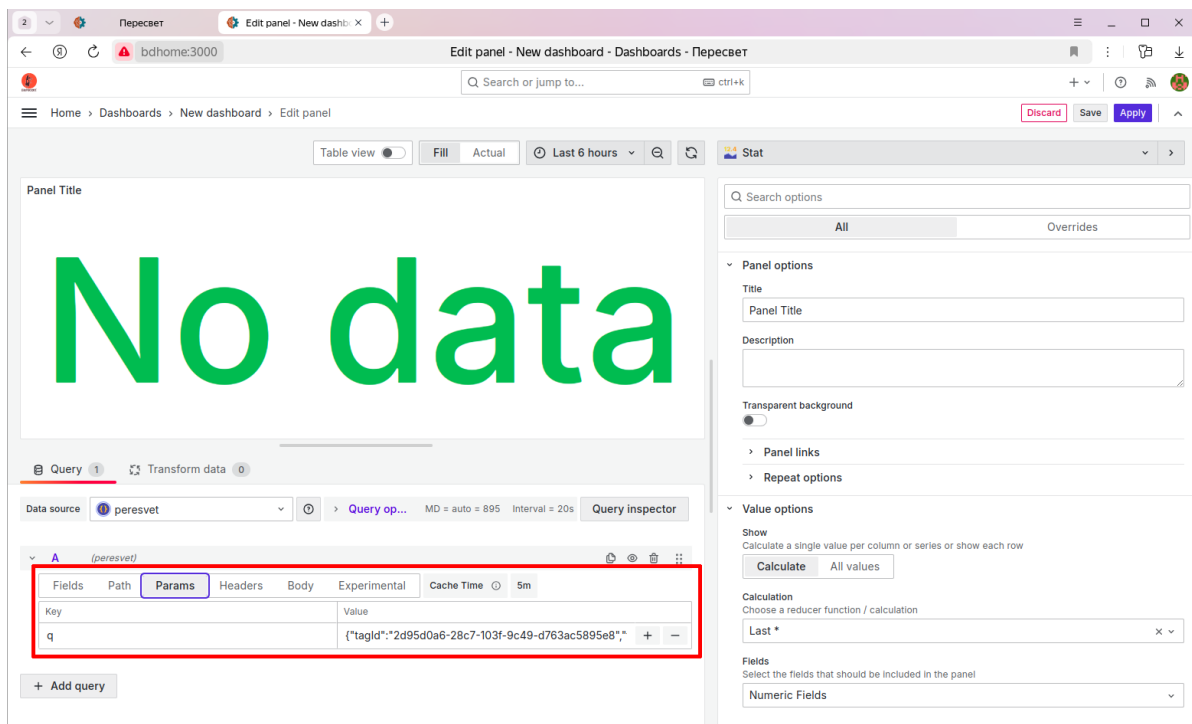
4. Настроим запросы получения данных.

Для того, чтобы не писать запрос вручную, его можно скопировать из конфигуратора.

Открываем конфигуратор, выбираем тег «I» и копируем запрос получения текущего значения тега.



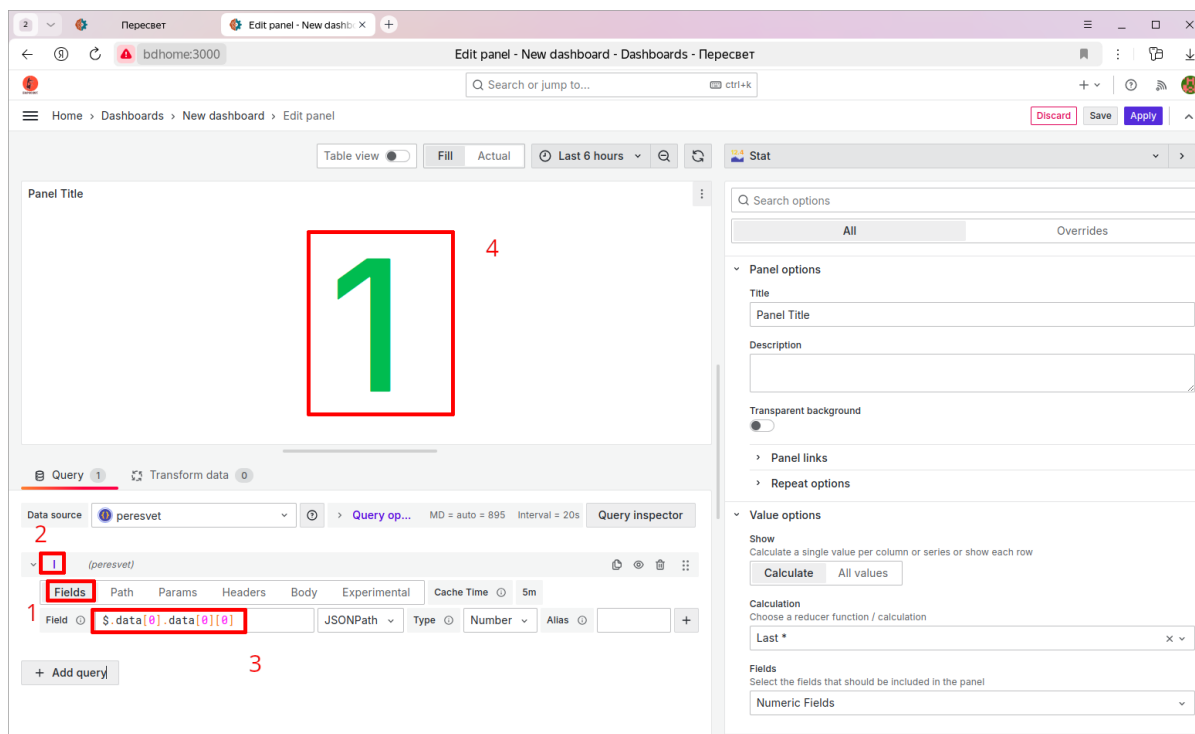
В окне настройки панели выбираем закладку «Params» и добавляем новый ключ «q». Значение ключа - скопированный нами запрос:



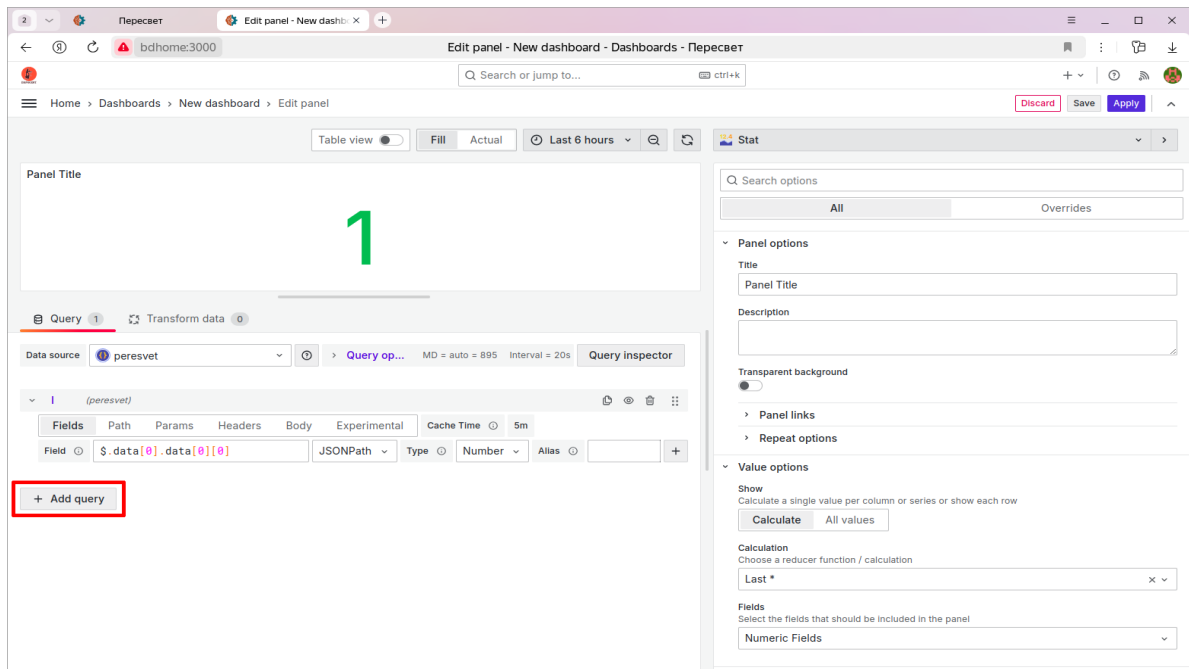
Далее:

1. Переходим на закладку «Fields»;

2. Меняем имя очереди на «I»;
3. Вводим выражение на языке JSONPath, разбирающее ответ от платформы с текущим значением тега
4. На экране появится «1» - текущее, введённое нами на предыдущих шагах значение тока.

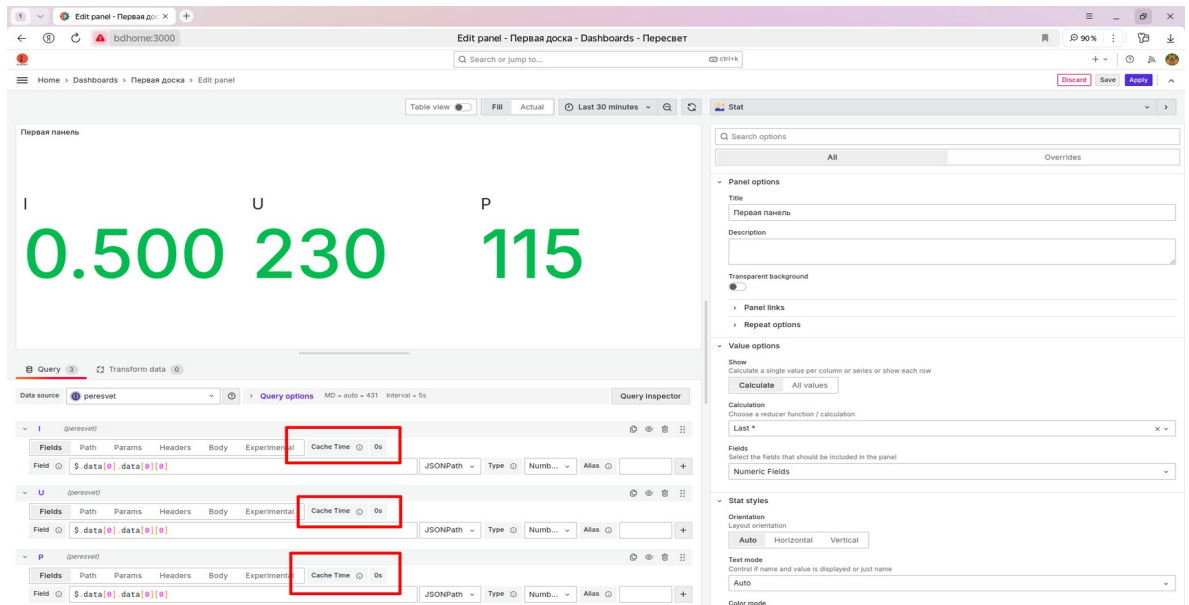


Повторим предыдущие шаги для оставшихся двух тегов напряжения и мощности, добавляя новые запросы нажатием кнопки «Add query»:



Предупреждение:

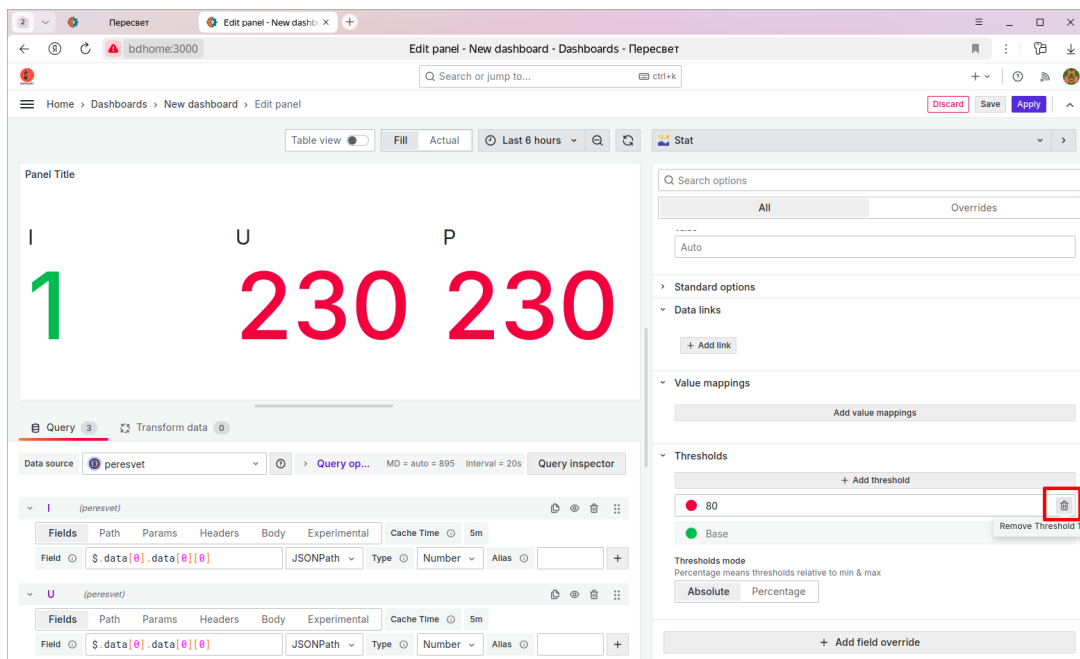
У каждого поля отменим кэширование данных:



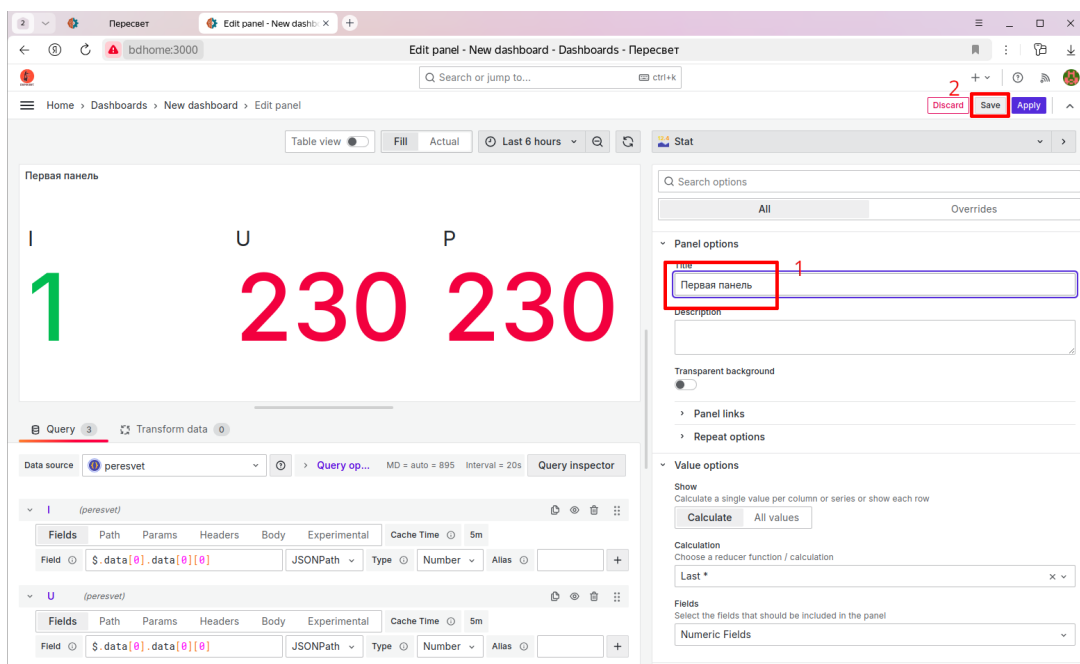
## 5. Окончательная настройка панели.

После настройки данных значения напряжения и мощности отображаются красным цветом. Так выставлены настройки по умолчанию.

1. Чтобы все значения отражались зелёным цветом, уберём порог значения «80»;



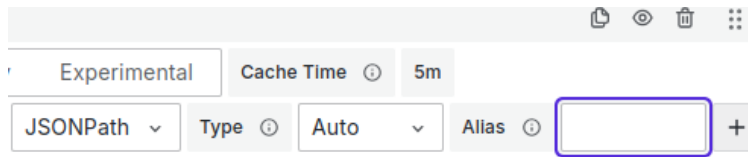
2. Переименуем панель и нажмем кнопку «Save»:



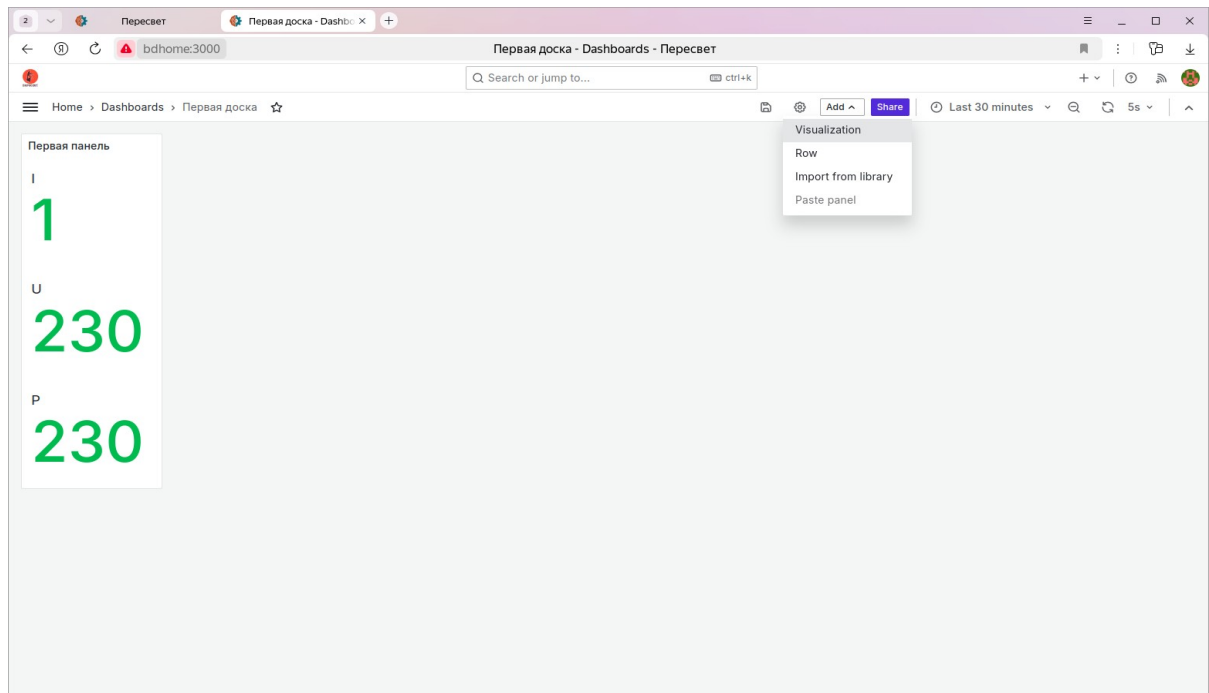
Появится окно, в котором нужно задать имя доски данных.

Предупреждение:

Чтобы на панели рядом с именем тега не отображался 0, введите в поля **Alias** пробел:



6. Выравниваем новую панель на экране и создадим новую панель, с трендом, для чего выберем команду «Add → Visualization»:



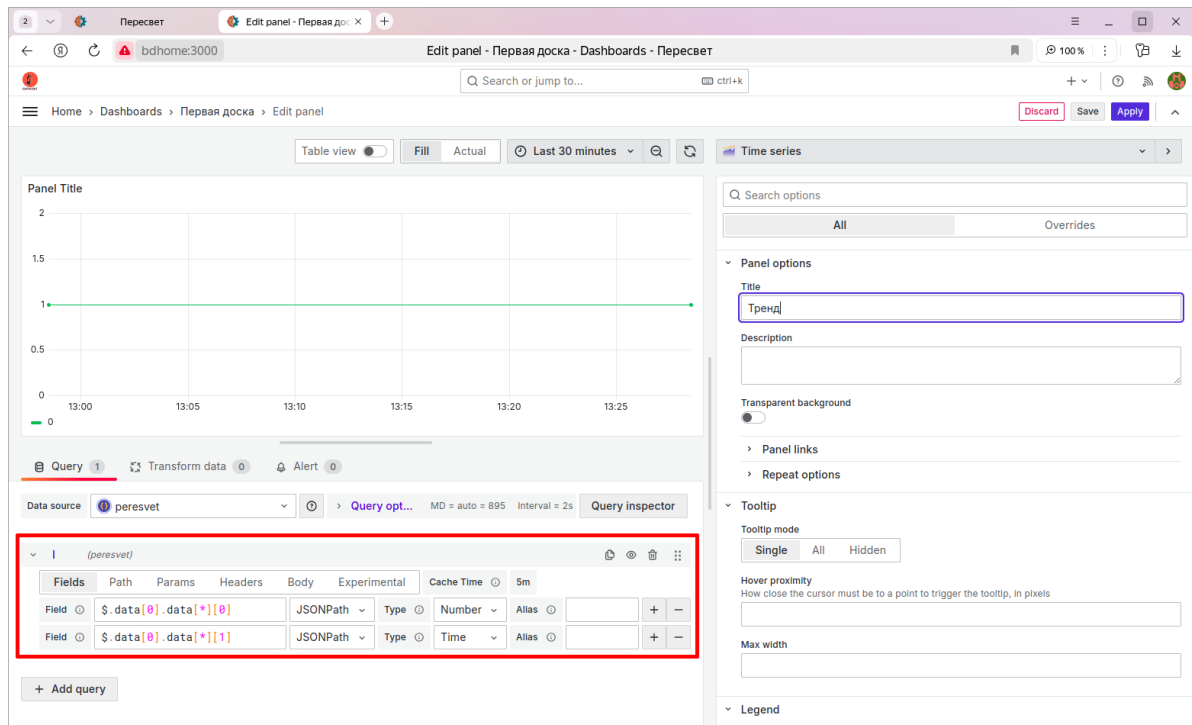
7. Настройка панели с трендом.

Тип панели оставим по умолчанию - «Time series».

Получение данных настроим похоже предыдущей панели, за исключением того, что ключ запроса «q» будет включать функции Grafana'ы `$__isoFrom()` и `$__isoTo()`, обозначающие начало и конец запрашиваемого временного периода:

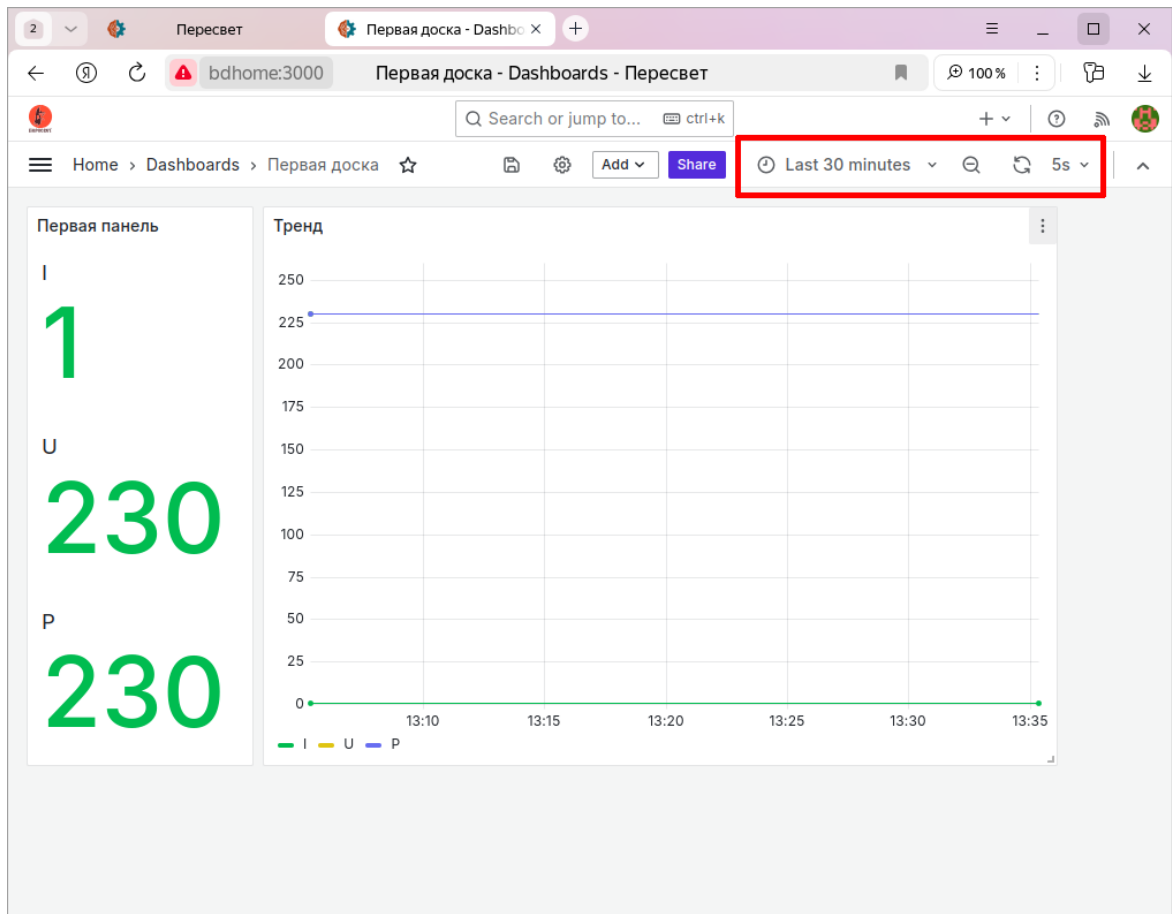
```
q={"tagId":"2d95d0a6-28c7-103f-9c49-d763ac5895e8","format":true,"start":"$__isoFrom()","finish":"$__isoTo()"}.
```

Закладки «Fields» у всех трёх запросов должны быть такого вида:



## 8. Выбор временных периодов.

Настроив панель с трендом для отображения трёх тегов, приведём доску данных приблизительно к такому виду и выберем диапазон времени для отображения на экране, а также периодичность обновления данных:



9. Проверим работу модели. Для чего записываем новые значения в теги «I» и «U» и наблюдаем изменение данных на панели с электрическими параметрами:

