

АКЦИОНЕРНОЕ ОБЩЕСТВО  
«ЦИФРОВЫЕ ПЛАТФОРМЫ И РЕШЕНИЯ УМНОГО ГОРОДА»

ПЭВМ «Отраслевая промышленная IoT платформа»  
(альтернативное название ПЭВМ «Инфраструктурная IoT платформа»)

**Руководство администратора**

Листов: 42

Москва

2021

## Содержание

1.	Введение.....	5
1.1.	Термины, определения и сокращения.....	5
2.	Общие положения.....	6
2.1.	Наименование.....	6
2.2.	Перечень эксплуатационной документации.....	6
3.	Назначение разработки.....	7
3.1.	Назначение документа.....	7
3.2.	Цели разработки.....	7
3.3.	Требования к оборудованию и программному обеспечению.....	7
3.3.1.	Требования к стационарным рабочим местам.....	7
3.3.2.	Требования к серверному оборудованию.....	7
3.3.3.	Состав программного обеспечения.....	7
4.	Подготовка к работе.....	8
4.1.	Состав и структура дистрибутива.....	8
4.2.	Развертывание Платформы.....	8
5.	Контроль доступа на основе ролей для устройств и приложений.....	8
6.	Практическая работа администратора.....	9
6.1.	Панель «Владельцы».....	9
6.2.	Панель «Администраторы владельца».....	11
6.2.1.	Вход в роли администратора владельца.....	11
6.2.2.	Добавление администратора владельца.....	12
6.3.	Поиск в списке администраторов владельца.....	13
7.	Администрирование клиента.....	14
7.1.	Пример администрирования клиента с профилем «Школа».....	14
7.1.1.	Перечень клиентов.....	14
7.1.1.	Управление активами клиента.....	15
7.1.2.	Управление Цепочками клиента.....	16
7.1.3.	Управление устройствами клиента.....	17
7.1.4.	Устройство: аутентификация на основе токена доступа.....	17
7.1.5.	Управление дашбордами клиента.....	18
7.1.6.	Удаление клиента.....	18
7.1.7.	Вход в роли «пользователь клиента».....	18
7.1.8.	Активы клиента (просмотр пользователем).....	19
7.1.9.	Устройства клиента(просмотр пользователем).....	19
8.	Администрирование цепочек правил.....	20

8.1.	Исполняющая система (ИС) .....	20
8.1.	Информационные структуры Исполняющей системы .....	20
8.2.	Корневая цепочка( <i>root rule chain</i> ) .....	21
8.3.	Обзор механизма правил.....	21
8.3.1.	Сообщение механизма правил .....	21
8.3.2.	Узел правила .....	22
8.3.3.	Связь узла правила.....	22
8.3.4.	Цепочка правил.....	22
8.3.5.	Результат обработки сообщения .....	23
8.4.	Типичные варианты использования .....	23
8.5.	Очередь механизма правил.....	24
8.5.1.	Стратегия отправки очереди .....	25
8.5.2.	Стратегия обработки очереди .....	25
8.5.3.	Очереди по умолчанию .....	26
9.	Библиотека встроенных узлов - примеры.....	27
9.1.	Узлы фильтрации .....	27
9.1.1.	Check Relation Filter Node.....	27
9.1.2.	Check Existence Fields Node.....	27
9.1.3.	Message Type Filter Node .....	27
9.2.	Узлы обогащения .....	28
9.2.1.	Calculate delta .....	28
9.2.2.	Customer attributes.....	29
9.3.	Узлы трансформации .....	29
9.3.1.	Change originator .....	29
9.3.2.	Script Transformation Node.....	30
10.	Практическая работа с цепочками правил.....	30
10.1.	Вход в систему.....	30
10.2.	Главное меню.....	31
10.3.	Панель «Цепочки правил».....	31
10.4.	Просмотр цепочки .....	32
10.5.	Экспорт цепочки в формате json.....	32
10.6.	Создание и редактирование цепочки.....	33
10.6.1.	Создание новой цепочки .....	33
10.6.2.	Переименование новой цепочки .....	34
10.6.3.	Добавление узлов и связей .....	34
10.6.4.	Сохранение изменений .....	37

10.6.5.	Тестирование JS- скрипта.....	37
11.	Примеры решения прикладной задачи .....	40
11.1.	Root Rule Chain .....	40
11.2.	Расчет КПД УРЭ Водонапорной станции .....	40
11.3.	Расчет небалансовых потерь .....	41
11.4.	Расчет потери воды в сети водоснабжения.....	41
11.5.	Генератор .....	42

## 1. Введение

Настоящий документ предоставляет сведения о ПЭВМ «Отраслевая промышленная IoT платформа» (альтернативное название «Инфраструктурная IoT платформа»), необходимые администратору системы.

Документ разработан в соответствии с требованиями стандарта ГОСТ 34.201-89 «Комплекс стандартов на автоматизированные системы «Виды, комплектность и обозначение документов при создании автоматизированных систем».

### 1.1. Термины, определения и сокращения

Термин/ Сокращение	Определение
Платформа, Система	ПЭВМ «Инфраструктурная IoT платформа»/ ПЭВМ «Отраслевая промышленная IoT платформа»
Интернет вещей	Концепция сети передачи данных между физическими объектами («вещами»), оснащёнными встроенными средствами и технологиями для взаимодействия друг с другом или с внешней средой.
MQTT	Упрощённый сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщениями между устройствами по принципу издатель-подписчик.
CoAP	Специализированный протокол интернет-приложений для устройств с ограниченными возможностям. Это позволяет этим ограниченным устройствам, называемым "узлами", взаимодействовать с более широким Интернетом, используя аналогичные протоколы.
Цепочка правил	Описывает процесс обработки входящих сообщений.
ИС	Исполняющая система цепочек правил на платформе. ИС определяет всю логику работы системы. На основе

	встроенного механизма событий выполняет процессы, описываемые цепочками.
--	--

## 2. Общие положения

ПЭВМ «Инфраструктурная IoT платформа» предназначена для работы в качестве отраслевого решения в области автоматизированных систем управления технологическими процессами (АСУ ТП). Основные задачи Платформы - мониторинг и управление технологическим оборудованием в режиме реального времени, работа с устройствами автоматики, контроллерами и т.д., аналитическая обработка полученных данных для определения загруженности и эффективности использования промышленного оборудования, учет потребления энергоресурсов на предприятиях, расчет ключевых показателей энергоэффективности.

### 2.1. Наименование

Полное наименование ПО - «Инфраструктурная IoT платформа».

Альтернативное наименование ПЭВМ «Отраслевая промышленная IoT платформа»

Условное наименование ПО - Платформа.

### 2.2. Перечень эксплуатационной документации

Настоящий документ входит в состав комплекта документации технического проекта, включающего следующие документы:

- «IoT платформа -Руководство\_Пользователя»;
- «IoT платформа - Руководство\_Администратора».

### 3. Назначение разработки

#### 3.1. Назначение документа

Руководство администратора предназначено для обеспечения порядка функционирования ПЭВМ «Инфраструктурная IoT платформа». В документе указаны все требования для установки, развертывания и поддержания работоспособности Платформы.

#### 3.2. Цели разработки

Цель создания Платформы - мониторинг и управление технологическим оборудованием в режиме реального времени, работа с устройствами автоматизации, контроллерами и т.д.

#### 3.3. Требования к оборудованию и программному обеспечению

##### 3.3.1. Требования к стационарным рабочим местам

Конечный пользователь может использовать для работы с Платформой рабочую станцию, имеющую выход в интернет с установленным браузером современной версии.

##### 3.3.2. Требования к серверному оборудованию

Минимальные требования к аппаратному обеспечению для установки Платформы указаны ниже (см. Таблица 1):

**Таблица 1. Требования к аппаратному обеспечению**

#	Назначение	Память, Гб	Ядра, шт.	Диск, Гб
1	Сервер Платформы	16	8	300

##### 3.3.3. Состав программного обеспечения

Для обеспечения функционирования ПО используется бесплатно распространяемое программное обеспечение с открытым исходным кодом. Состав используемого программного обеспечения системы приведен в ниже (см. Таблица 2)

**Таблица 2. Состав используемого программного обеспечения.**

№ п/п	Класс ПО	Наименование ПО и версия	Правообладатель	Лицензия	Кол-во
1.	Операционная система.	Ubuntu, версия 20.x	Canonical Ltd.	GNU GPL	1
2.	Система управления базами данных.	PostgreSQL, версия 13.2	The PostgreSQL Global Development Group.	PostgreSQL license	1

№ п/п	Класс ПО	Наименование ПО и версия	Правообладатель	Лицензия	Кол-во
3.	Система обмена сообщениями.	Apache Pulsar Версия 2.8.0	Apache Software Foundation.	Apache License	1

## 2.5. Требование к квалификации администратора

Администратор должен обладать следующими навыками:

- опыт в работе с Microsoft Windows 7 и выше;
- опыт администрирования ОС Linux (Debian / Ubuntu);
- опыт настройки рабочих станций локальной вычислительной сети;
- опыт решения вопросов инсталляции, общесистемного сопровождения и администрирования;
- опыт работы в области администрирования СУБД;
- опыт работы с системой контейнерной виртуализации (docker).

## 4. Подготовка к работе

### 4.1. Состав и структура дистрибутива

ПО Платформы поставляется в виде образа виртуальной машины, доступный для скачивания из хранилища Amazon S3.

### 4.2. Развертывание Платформы

Процесс развертывания подробно описан в документе «IoT\_Платф - Руководство по развертыванию демонстрационного стенда.docx»

## 5. Контроль доступа на основе ролей для устройств и приложений

Платформа поддерживает модель безопасности с тремя основными ролями:

1. Системный администратор;
2. Администратор владельца;
3. Пользователь клиента.



Системный администратор может управлять Владельцами, в то время как администратор владельца управляет устройствами, панелями мониторинга, клиентами и другими объектами, принадлежащими конкретному клиенту. Пользователь-клиент может просматривать информационные панели и устройства управления, назначенные конкретному клиенту. Функциональности достаточно для множества простых случаев использования, особенно для создания информационных панелей для конечных пользователей в реальном времени.

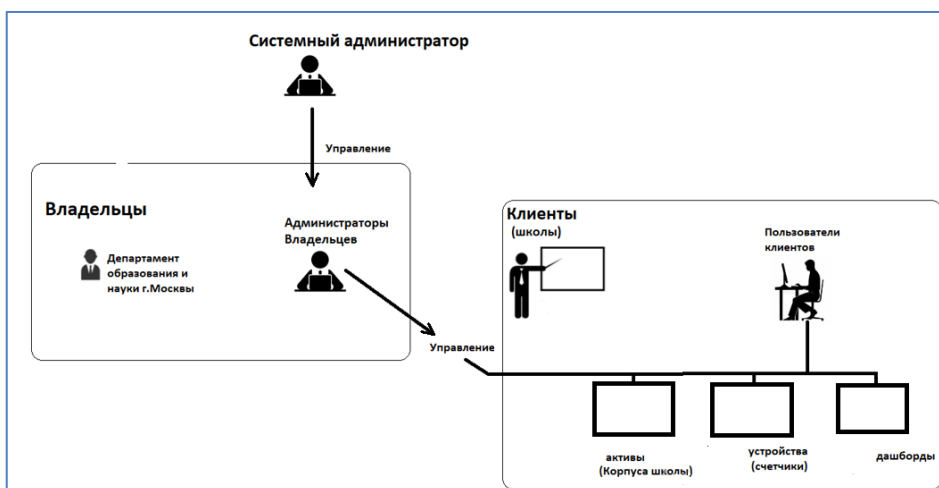


Рисунок 1. Пример использования модели с тремя ролями для Департамента городского образования.

## 6. Практическая работа администратора

### 6.1. Панель «Владельцы»

Панель «Владельцы» доступна для роли Системный администратор. На панели представлено меню и перечень владельцев в виде таблицы (см. Рисунок 2).

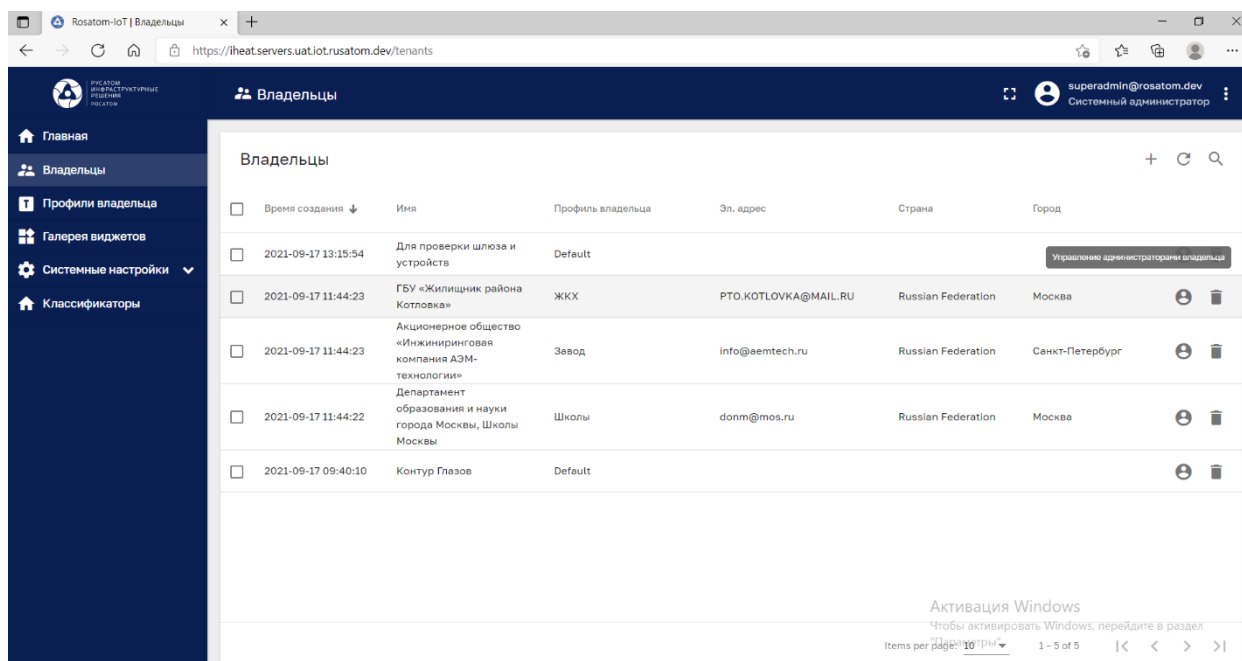



Рисунок 2. Перечень владельцев.

Для Управление администраторами владельца необходимо нажать на пиктограмму  в строке интересующего владельца.

## 6.2. Панель «Администраторы владельца»

Панель «Администраторы владельца» доступна для роли Системный администратор. На панели представлено меню и перечень администраторов владельца в виде таблицы (см. Рисунок 3).

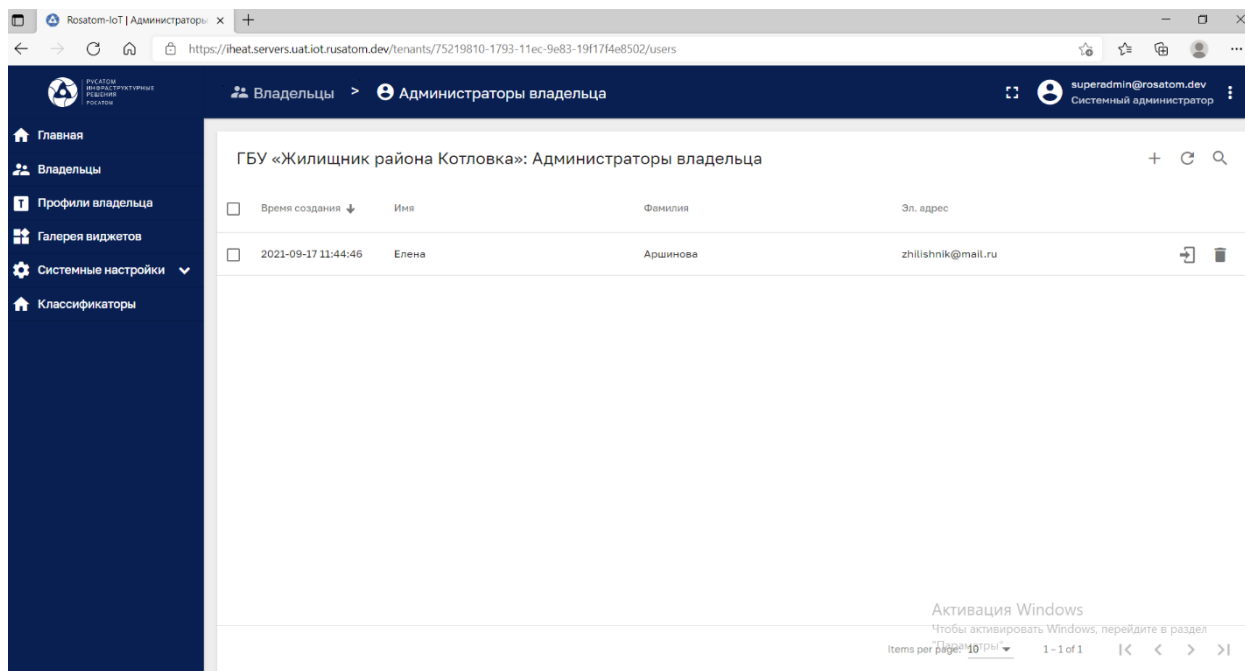

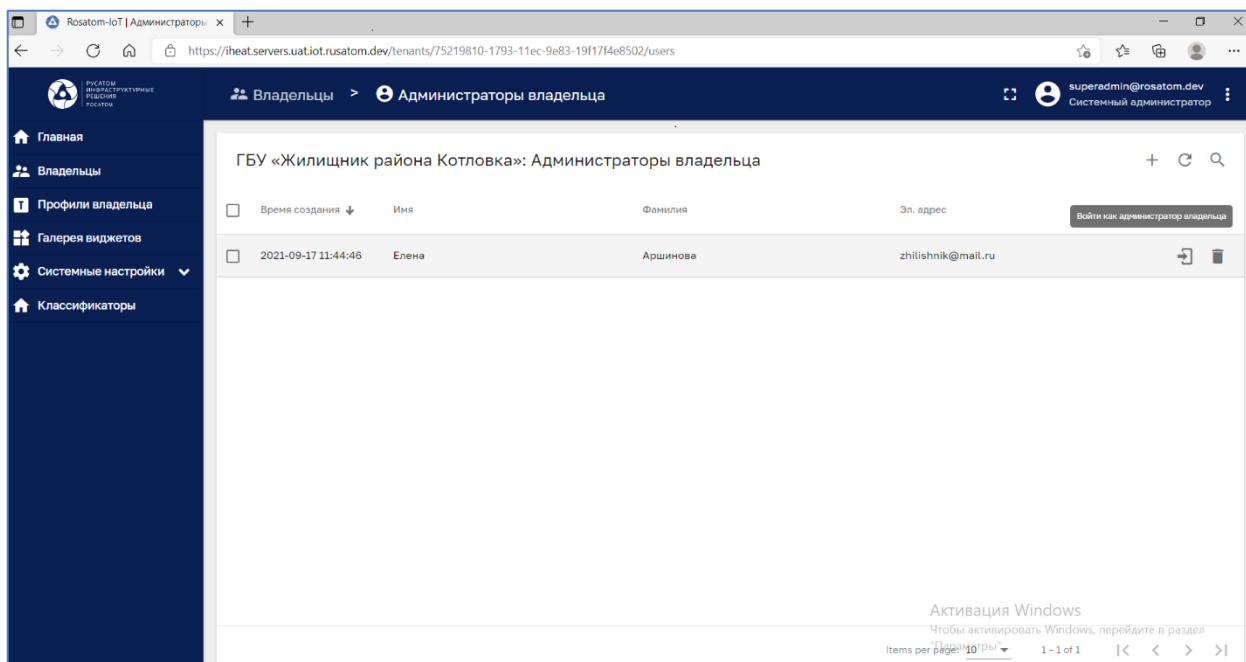


Рисунок 3. Панель «Администраторы владельца»

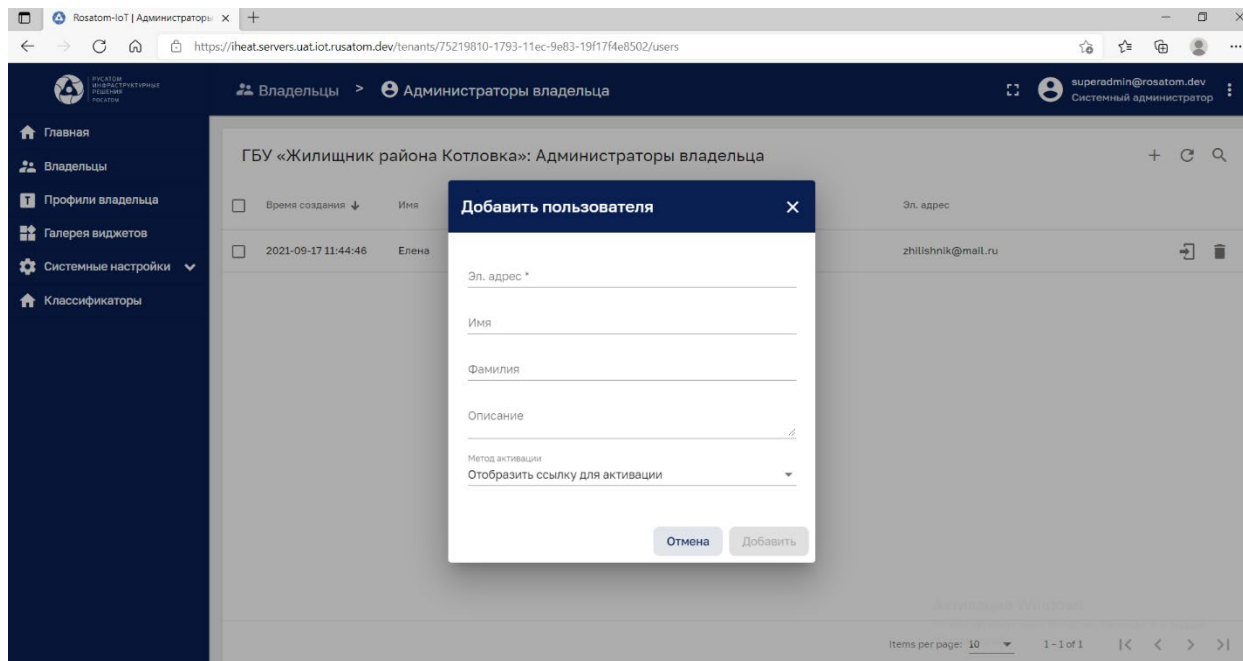
### 6.2.1. Вход в роли администратора владельца

Используя пиктограмму  системный администратор может войти в систему в роли администратора владельца (см. Рисунок 4).



**Рисунок 4. Вход в систему роли администратор владельца.****6.2.2. Добавление администратора владельца**

Для добавления администратора владельца системный администратор может нажать на кнопку «+», заполнить форму и нажать «добавить» (см. Рисунок 5).

**Рисунок 5. Добавление администратора владельца.**

### 6.3. Поиск в списке администраторов владельца

Для поиска в списке администраторов владельца необходимо щелкнуть пиктограмму «лупа» и ввести поисковый образец (см. Рисунок 6).

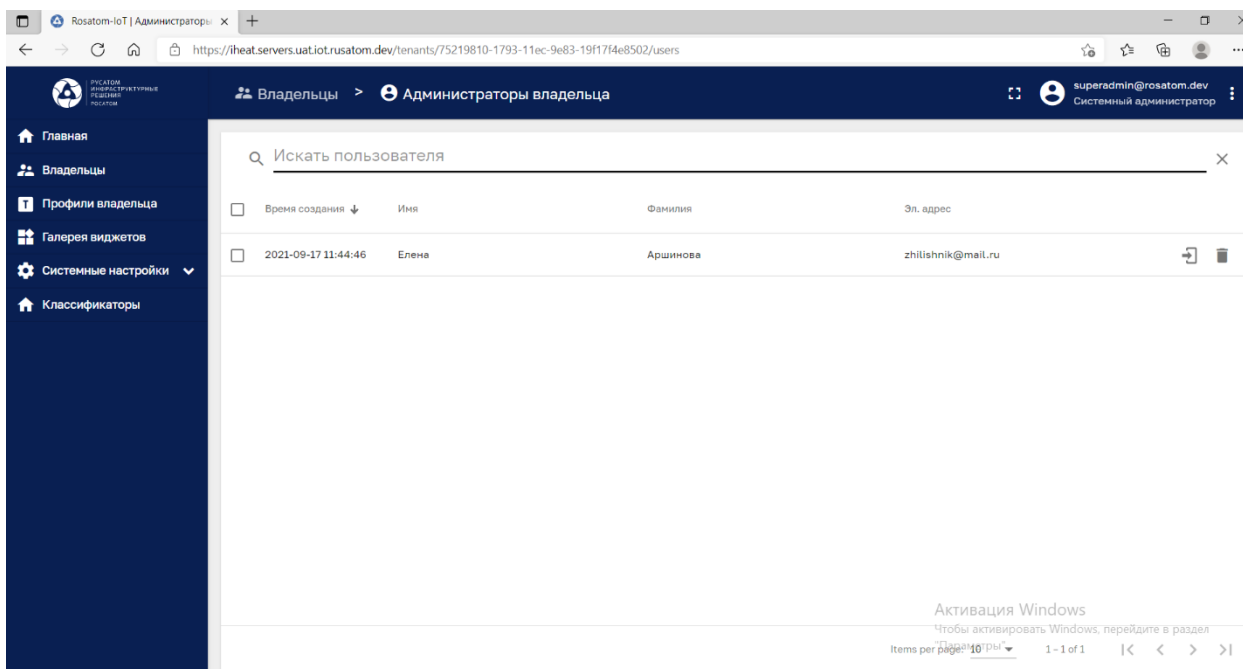


Рисунок 6. Поиск в списке администраторов владельца.

## 7. Администрирование клиента

Администрирование клиента и его объектов (активов, устройств, панелей) доступно для роли «Администратор владельца».

### 7.1. Пример администрирования клиента с профилем «Школа»

Главное меню для администрирования клиентских объектов показано ниже (см. Рисунок 7).

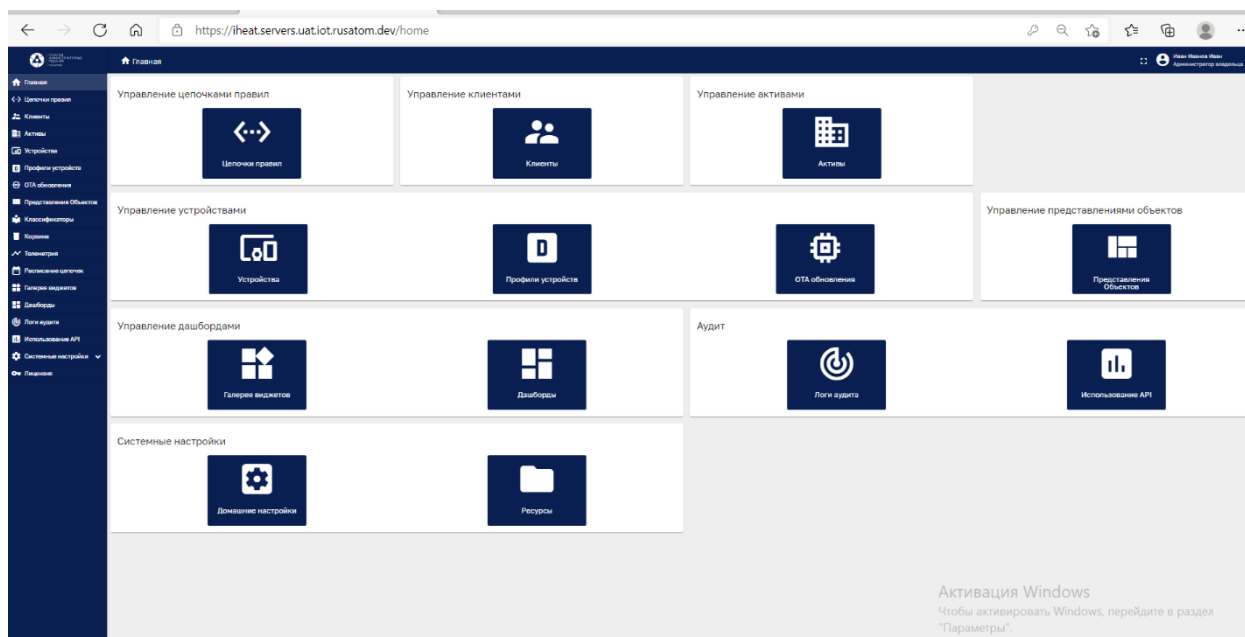


Рисунок 7. Меню клиентских объектов.

#### 7.1.1. Перечень клиентов

Перечень клиентов имеет вид таблицы. В каждой строке есть набор пиктограмм:

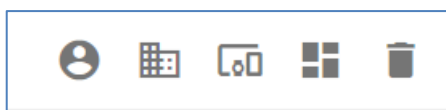


Рисунок 8.

Пиктограммы предназначены для:

- Управление пользователями клиента;
- Управление активами клиента;
- Управление устройствами клиента;
- Управление дашбордами клиента;
- Удаления клиента.

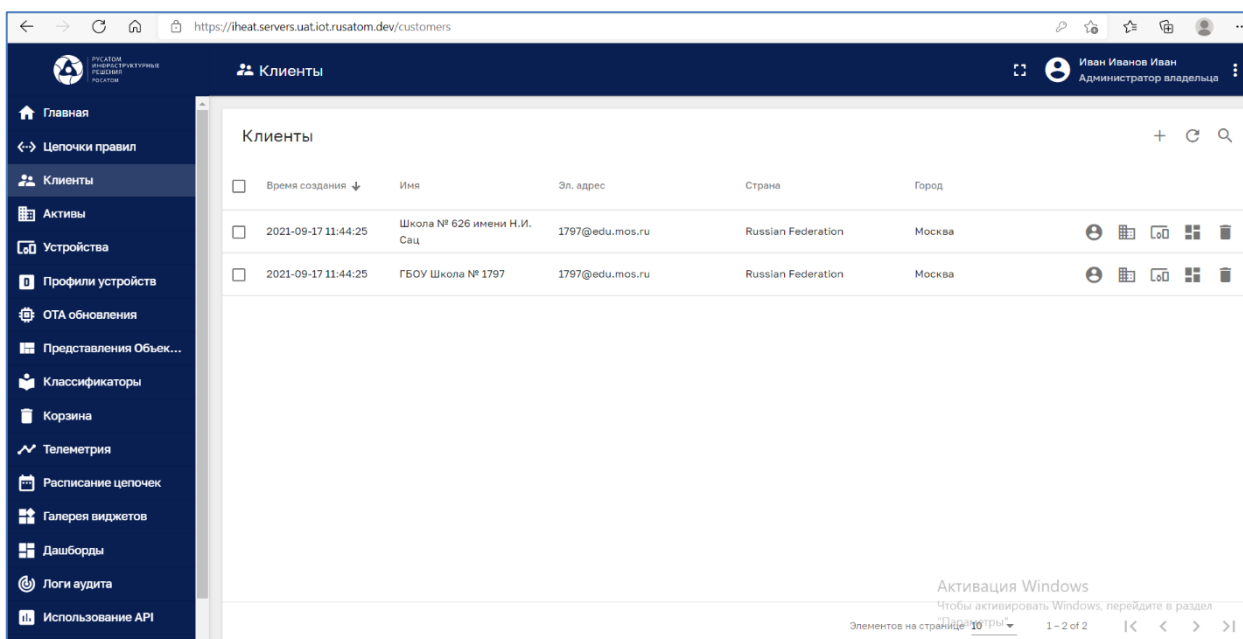


Рисунок 9. Панель управления клиентами и клиентскими объектами.

### 7.1.1. Управление активами клиента

Для управления активами в Главном меню для администрирования клиентских объектов необходимо нажать на кнопку «Активы»(см. Рисунок 10).

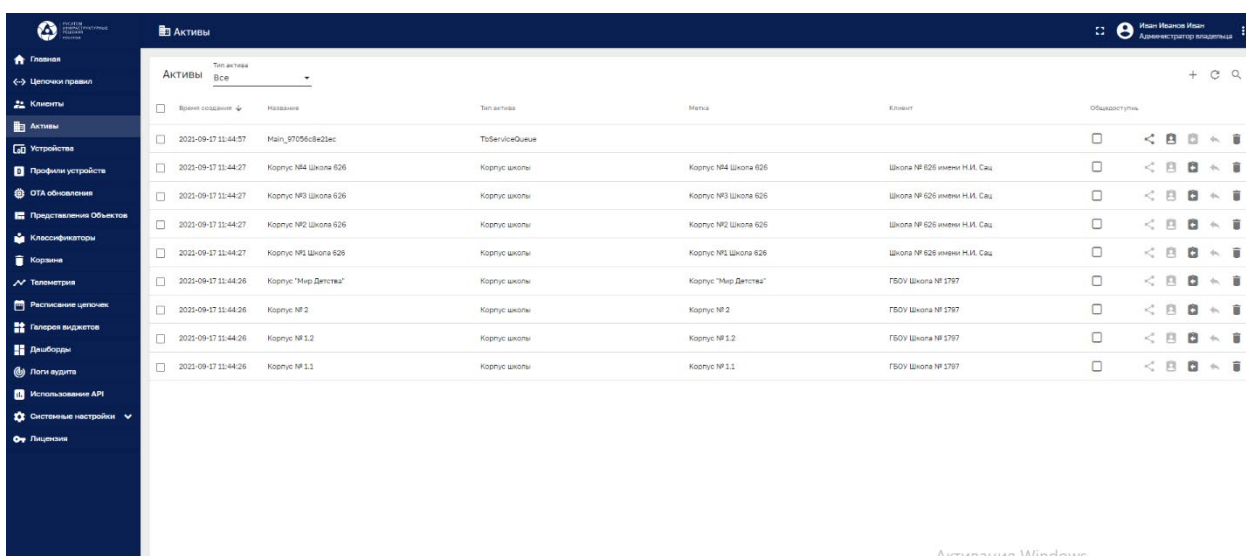


Рисунок 10. Панель «Активы»

В каждой строке перечня активов есть набор меню:



Для выполнения операций над активом:

- Предоставить Общий доступ;
- Присвоить;
- Отозвать;
- <...>
- Удалить.

### 7.1.2. Управление Цепочками клиента

Для управления цепочками в Главном меню для администрирования клиентских объектов необходимо нажать на кнопку «Цепочки правил»(см. Рисунок 11).

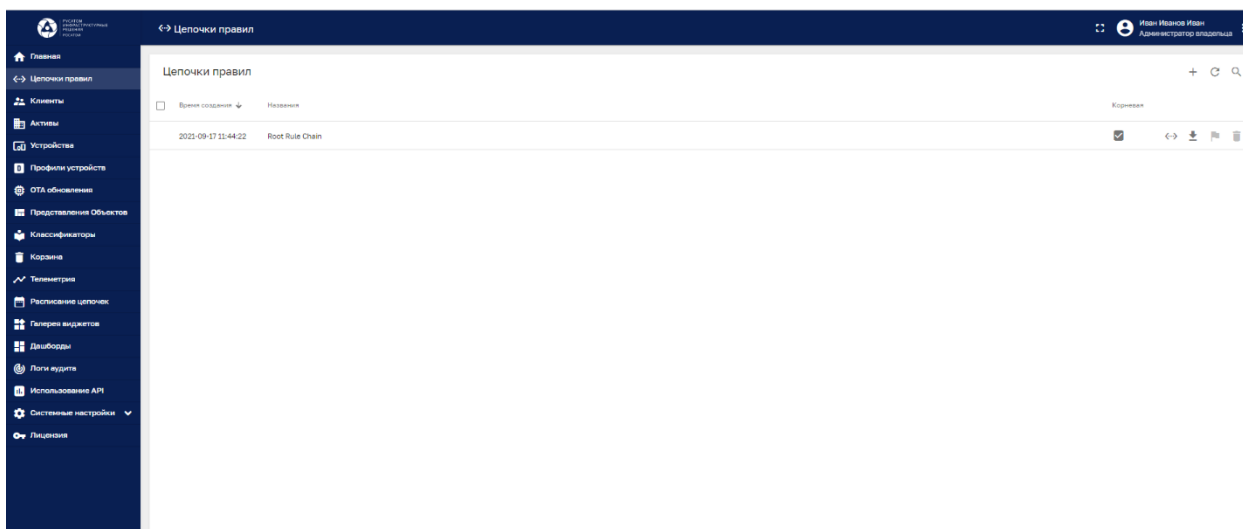



Рисунок 11. Панель «цепочки правил».

Применение пиктограммы  в строке выбранной цепочки открывает панель редактора цепочек (см. Рисунок 12):

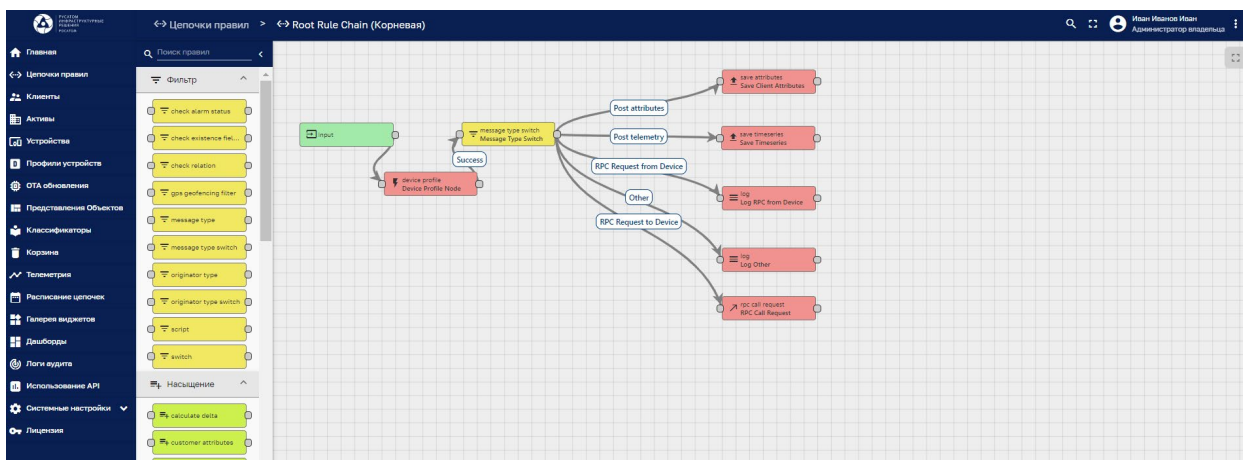


Рисунок 12. Интерфейс редактора цепочек



### 7.1.3. Управление устройствами клиента

Для управления устройствами клиента в Главном меню для администрирования клиентских объектов необходимо нажать на кнопку «Устройства»(см. Рисунок 13).

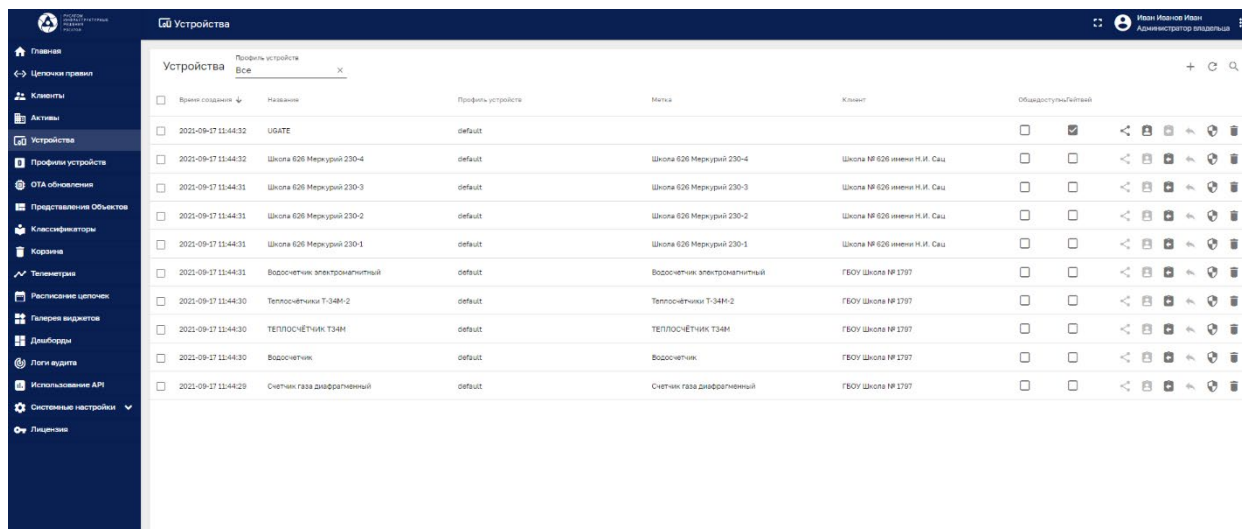



Рисунок 13. Панель «Устройства клиента»

Управление устройством выполняется при помощи меню в строке устройства:



### 7.1.4. Устройство: аутентификация на основе токена доступа

Если нажать пиктограмму  «управление учетными данными», то можно узнать и сохранить «токен доступа» (см. Рисунок 14):

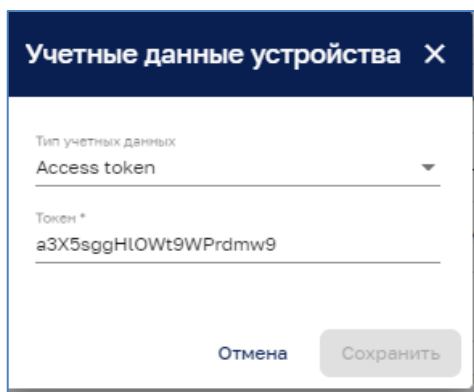
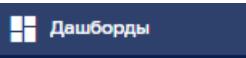


Рисунок 14. Токен доступа.


Аутентификация на основе токена доступа - это тип аутентификации устройства по умолчанию. После создания устройства в системе создается токен доступа по умолчанию.

Впоследствии его можно будет изменить. Чтобы подключить устройство к серверу с использованием аутентификации на основе токена доступа, клиент должен указать токен доступа как часть URL-адреса запроса (для HTTP и CoAP) или как имя пользователя в сообщении подключения MQTT.


### 7.1.5. Управление дашбордами клиента

Для управления дашбордами в Главном меню для администрирования клиентских объектов необходимо нажать на кнопку .

### 7.1.6. Удаление клиента

Для удаления клиента используется кнопка  в строке клиента на панели «Клиенты».

### 7.1.7. Вход в роли «пользователь клиента»

На панели Пользователя клиента Администратор владельца может войти в систему как Пользователь клиента. Для этого нужно воспользоваться пиктограммой  (см. Рисунок 15):

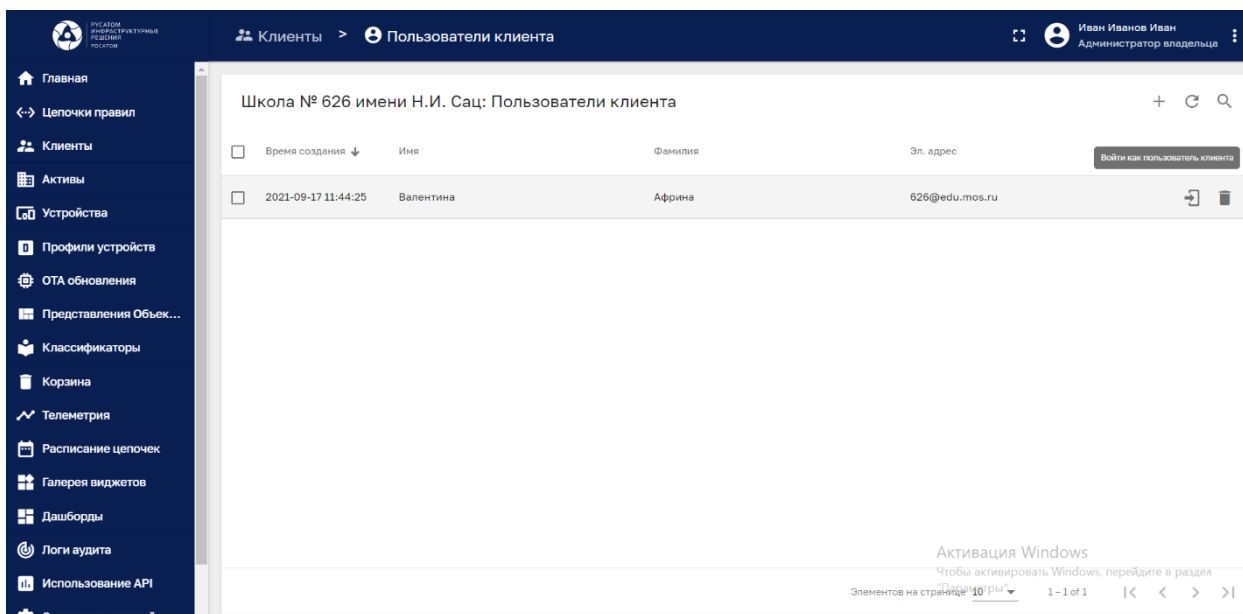


Рисунок 15. Вход в систему в роли Пользователь клиента.

### 7.1.8. Активы клиента (просмотр пользователем)

Просмотр активов клиента доступен для роли «пользователь клиента» (см. Рисунок 16).

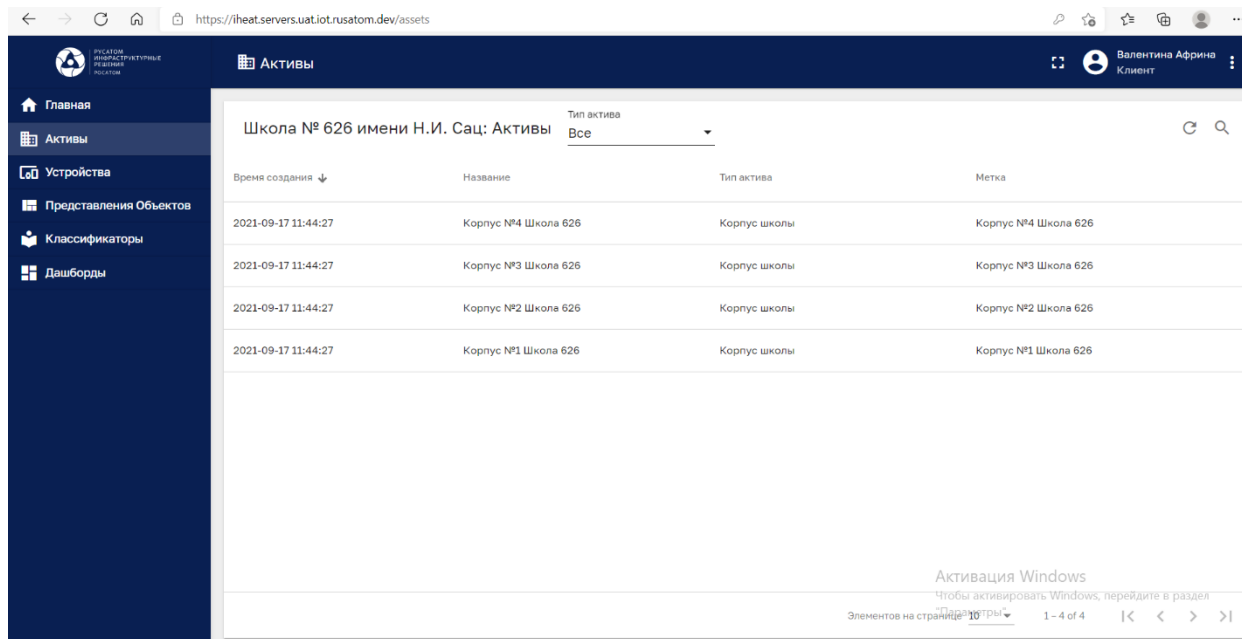


Рисунок 16. Активы клиента.

### 7.1.9. Устройства клиента(просмотр пользователем)

Просмотр устройств клиента доступен для роли «пользователь клиента» (см. Рисунок 17)<sup>1</sup>.

---

1

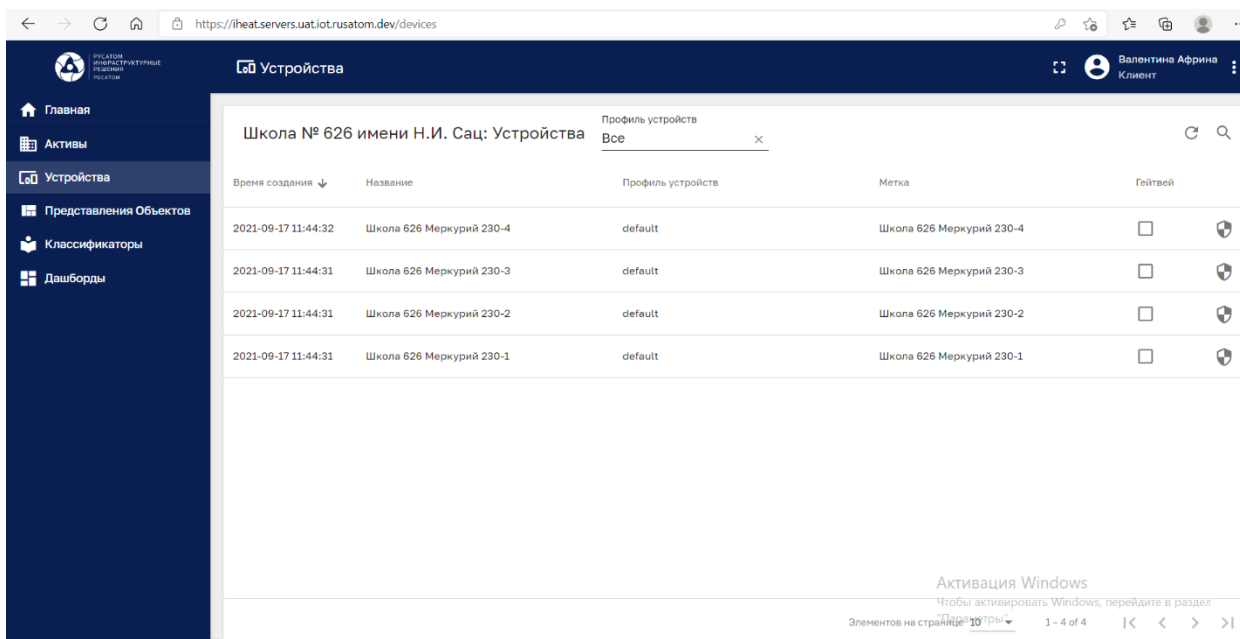


Рисунок 17. Устройства клиента.

## 8. Администрирование цепочек правил

### 8.1. Исполняющая система (ИС)

Исполняющая система (далее -ИС) - это интерпретирующая система для цепочек правил. ИС определяет всю логику работы системы. ИС имеет встроенный механизм обработки событий и выполняет процессы, описываемые цепочками.

ИС может настраивается, конфигурироваться и обеспечивает ряд функций:

- Фильтрацию;
- Обогащение данных;
- Передачу сообщений.

### 8.1. Информационные структуры Исполняющей системы

ИС обрабатывает 3 основных вида информационных структур:

1. Сообщение - любое входящее событие. Это могут быть входящие данные от устройств, событие жизненного цикла устройства, событие REST API, запрос RPC и т. Д.
2. Узел правила - функция, которая выполняется для входящего сообщения. Существует множество различных типов узлов, которые могут фильтровать, преобразовывать или выполнять какие-либо действия с входящим сообщением.

3. Цепочка правил - узлы связаны друг с другом отношениями. Исходящее сообщение от узла правила отправляется следующим подключенным узлам правила.

## 8.2. Корневая цепочка(*root rule chain*)

Имеется корневая цепочка, которая принимает все входные сообщения по умолчанию.

Корневая цепочка обозначается как *root rule chain*.

Вся обработка может быть описана в корневой цепочке, или передаваться далее в другие цепочки.

## 8.3. Обзор механизма правил

С помощью механизма правил можно фильтровать, обогащать и преобразовывать входящие сообщения, исходящие от устройств IoT и связанных активов. Можно также запускать различные действия, например, уведомления или осуществлять связь с внешними системами.

### 8.3.1. Сообщение механизма правил

Сообщение механизма правил - это неизменяемая структура данных, которая представляет различные сообщения в системе.

Например:

- Входящая телеметрия, обновление атрибутов или вызов RPC с устройства;
- Событие жизненного цикла объекта: создано, обновлено, удалено, назначено, не назначено, атрибуты обновлены;
- Событие состояния устройства: подключено, отключено, активно, неактивно и т.д.;
- Другие системные события.

Сообщение содержит следующую информацию:

- **Message ID**; Идентификатор сообщения: универсальный уникальный идентификатор на основе времени;
- **Originator**; Отправитель сообщения: устройство, объект или другой идентификатор объекта;
- **Message type**; Тип сообщения: «Почтовая телеметрия» или «Событие бездействия» и т.д.;
- **Payload**; Полезная нагрузка сообщения: тело в формате JSON;
- **Metadata**; Метаданные: список пар ключ-значение с дополнительными данными о сообщении.

### 8.3.2. Узел правила

Узел правил - это базовый компонент механизма правил, который обрабатывает одно входящее сообщение за раз и создает одно или несколько исходящих сообщений. Узел правил - это основная логическая единица механизма правил. Узел правил может фильтровать, обогащать, преобразовывать входящие сообщения, выполнять действия или взаимодействовать с внешними системами.

### 8.3.3. Связь узла правила

Узлы правил могут быть связаны с другими узлами правил. Каждое отношение имеет тип отношения, метку, используемую для определения логического значения отношения. Когда узел правила создает исходящее сообщение, он всегда указывает тип отношения, который используется для маршрутизации сообщения к следующим узлам.

Типичные отношения узла правила - «Успех» и «Неудача». Узлы правил, которые представляют логические операции, могут использовать «Истина» или «Ложь». Некоторые конкретные узлы правил могут использовать совершенно разные типы отношений, например: «Публикация телеметрии», «Атрибуты обновлены», «Созданная сущность».

### 8.3.4. Цепочка правил

Цепочка правил - это логическая группа узлов правил и их отношений. Например, приведенная ниже цепочка правил будет:

- сохранять все сообщения телеметрии в базу данных;
- поднимать «Аварийный сигнал высокой температуры», если в поле температуры в сообщении будет больше 50 градусов;
- поднимать «Аварийный сигнал низкой температуры», если в поле температуры в сообщении будет ниже -40 градусов;
- записывать в журнал сбой выполнения сценариев проверки температуры на консоль в случае логической или синтаксической ошибки в сценарии.

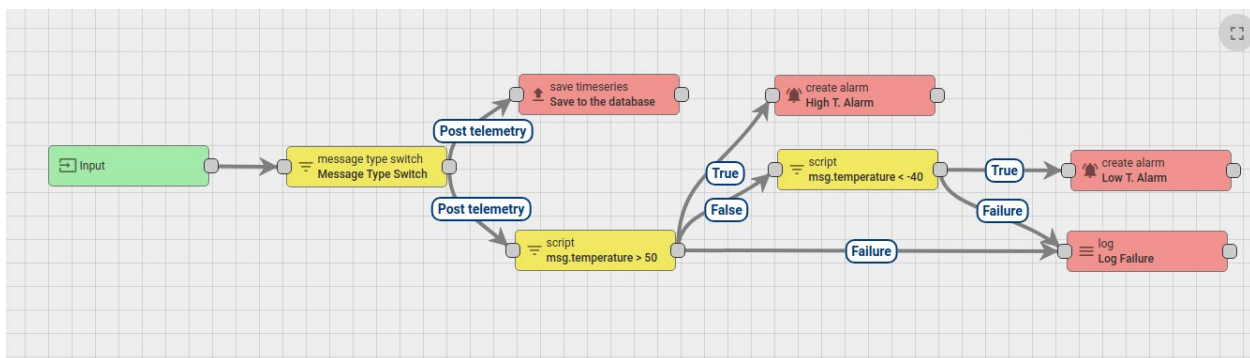


Рисунок 18.

Администратор системы может определить одну цепочку корневых правил и, при желании, несколько других цепочек правил. Цепочка корневых правил обрабатывает все входящие сообщения и может пересылать их в другие цепочки правил для дополнительной обработки. Другие цепочки правил также могут пересылать сообщения в разные цепочки правил.

Например, приведенная ниже цепочка правил будет:

- поднимать «Аварийный сигнал высокой температуры» если в поле температуры в сообщении будет больше 50 градусов;
- снимать флажок «Аварийный сигнал высокой температуры», если в поле температуры в сообщении будет меньше 50 градусов;
- пересылать события о тревогах «Создано» и «Сброшено» во внешнюю цепочку правил, которая обрабатывает уведомления для соответствующих пользователей.

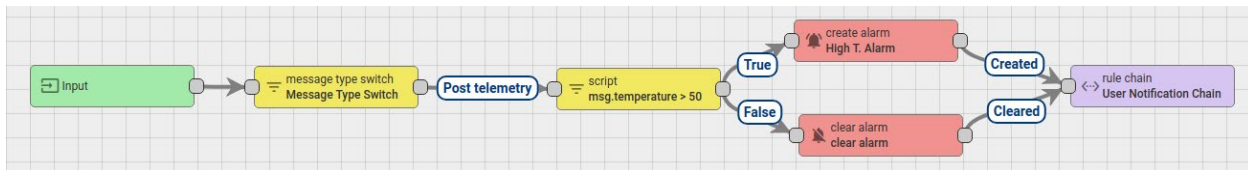


Рисунок 19.

### 8.3.5. Результат обработки сообщения

Возможны три результата обработки сообщения: успех, сбой и тайм-аут. Попытка обработки сообщения помечается как «Успешная», когда последний узел правила в цепочке обработки успешно обработал сообщение. Попытка обработки сообщения помечается как «Сбой», если один из узлов правил выдает «Сбой» обработки сообщения, и нет узлов правила, которые могли бы обработать этот сбой. Попытка обработки сообщения помечается как «Тайм-аут», когда общее время обработки превышает настраиваемый порог.

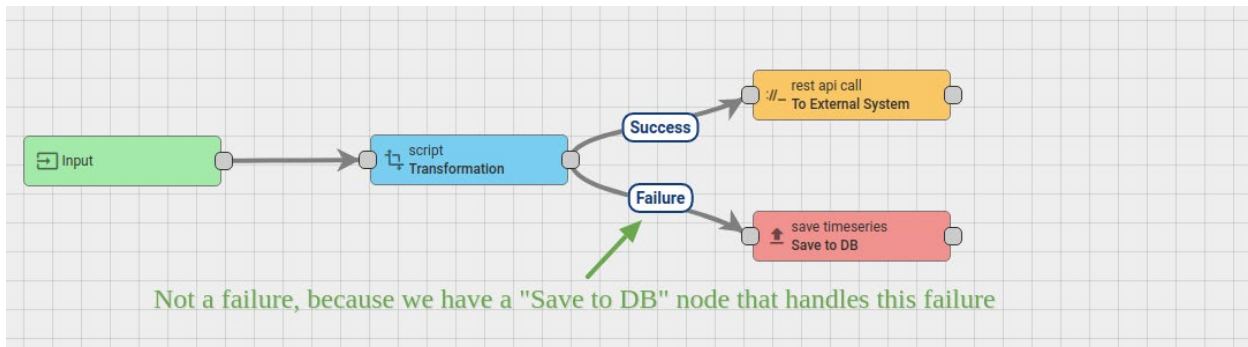


Рисунок 20.

## 8.4. Типичные варианты использования

Механизм правил - это настраиваемая среда для обработки сложных событий. Вот несколько распространенных вариантов использования, которые можно настроить с помощью цепочек правил:

- Проверка и изменение данных для входящей телеметрии или атрибутов перед сохранением в базе данных.

- Копирование данных телеметрии или атрибутов с устройств в связанные активы, чтобы можно было агрегировать данные телеметрии. Например, данные с нескольких устройств могут быть объединены в связанный актив.
- Создание / обновление / удаление сигналов тревоги на основе определенных условий.
- Триггерные действия на основе событий жизненного цикла устройства. Например, создать оповещения, если устройство находится в сети / не в сети.
- Загрузить дополнительные данные, необходимые для обработки. Например, пороговое значение температуры загрузки для устройства, которое определено в атрибуте Device's Customer или Tenant.
- Запуск вызовов REST API во внешние системы.
- Отправка электронных писем, когда происходит сложное событие, и использование атрибутов других сущностей в шаблоне электронной почты.
- Учет предпочтения пользователя при обработке событий.
- Вызовы RPC на основе определенного условия.
- Интеграция с внешними конвейерами, такими как Kafka, Spark, сервисы AWS и т. д.

## 8.5. Очередь механизма правил

Механизм правил просматривает очереди при запуске и когда опрашивает новые сообщения. Всегда есть «Основная» тема, которая используется как основная точка входа для новых входящих сообщений. Вы можете настроить несколько очередей, используя `thingsboard.yml` или переменные среды. После настройки вы можете поместить сообщение в другую тему, используя узел «Контрольная точка». Это автоматически подтверждает соответствующее сообщение в текущей теме.

Определение очереди состоит из следующих параметров:

- имя - используется для статистики и ведения журнала;
- тема - используется реализациями Queue для создания и потребления сообщений;
- интервал опроса - продолжительность в миллисекундах между опросами сообщений, если новых сообщений не поступило;
- разделы - количество разделов, которые нужно связать с этой очередью. Используется для масштабирования количества сообщений, которые могут обрабатываться параллельно;
- `pack-processing-timeout` - интервал в миллисекундах для обработки конкретного пакета сообщений, возвращаемого потребителем;
- `submit-strategy` - определяет логику и порядок отправки сообщений механизму правил.
- `processing-strategy` - определяет логику подтверждения сообщений.



### 8.5.1. Стратегия отправки очереди

Служба Rule Engine постоянно опрашивает сообщения по определенной теме, и как только потребитель возвращает список сообщений, он создает объект TbMsgPackProcessingContext. Стратегия отправки очереди контролирует, как сообщения из TbMsgPackProcessingContext отправляются в цепочки правил.

Доступны 5 стратегий:

1. BURST - все сообщения отправляются в цепочки правил в порядке их поступления.
2. BATCH - сообщения группируются в пакеты с помощью параметра конфигурации «queue.rule-engine.queues [queue index] .batch-size». Новая партия не отправляется, пока не будет подтверждена предыдущая партия.
3. SEQUENTIAL\_BY\_ORIGINATOR - сообщения отправляются последовательно внутри определенного объекта (отправителя сообщения). Новое сообщение, например, для устройства A не отправляется до тех пор, пока не будет подтверждено предыдущее сообщение для устройства A.
4. SEQUENTIAL\_BY\_TENANT - сообщения отправляются последовательно в пределах тенанта (владельца отправителя сообщения). Новое сообщение, например, для арендатора A не отправляется до тех пор, пока не будет подтверждено предыдущее сообщение для арендатора A.
5. ПОСЛЕДОВАТЕЛЬНО - сообщения отправляются последовательно. Новое сообщение не отправляется, пока не будет подтверждено предыдущее сообщение. Это делает обработку довольно медленной.

### 8.5.2. Стратегия обработки очереди

Стратегия обработки контролирует, как повторно обрабатываются сообщения с ошибкой или превышением времени ожидания.

Доступны 5 стратегий:

1. SKIP\_ALL\_FAILURES - просто игнорировать все сбои и таймауты. Приведет к «потере» сообщений. Например, если БД не работает, сообщения не будут сохраняться, но все равно будут помечены как «подтвержденные» и удалены из очереди. Эта стратегия создана в основном для обратной совместимости с предыдущими выпусками и средами разработки / демонстрации.
2. RETRY\_ALL - повторить все сообщения из пакета обработки. Если 1 из 100 сообщений завершится ошибкой, стратегия все равно повторно обработает (повторно отправит в Rule Engine) 100 сообщений.
3. RETRY\_FAILED - повторить все неудачные сообщения из пакета обработки. Если 1 из 100 сообщений завершится ошибкой, стратегия повторно обработает (повторно отправит в Rule Engine) только 1 сообщение. Сообщения с истекшим временем ожидания не обрабатываются повторно.

4. `RETRY_TIMED_OUT` - повторить все просроченные сообщения из пакета обработки. Если время ожидания 1 из 100 сообщений истечет, стратегия повторно обработает (повторно отправит в Rule Engine) только 1 сообщение. Неудачные сообщения не будут обрабатываться повторно.
5. `RETRY_FAILED_AND_TIMED_OUT` - повторить все неудачные и просроченные сообщения из пакета обработки.

Все стратегии «RETRY \*» поддерживают важные параметры конфигурации:

- `retries` - Количество повторов, 0 не ограничено
- процент отказов - пропустить повторную попытку, если количество отказов или тайм-аутов меньше X процентов сообщений;
- `pause-between-retries` - время ожидания в потоке потребителя перед повторными попытками в секундах;

### 8.5.3. Очереди по умолчанию

По умолчанию настроены три очереди: `Main`, `HighPriority` и `SequentialByOriginator`. Они различаются в зависимости от стратегии отправки и обработки. Как правило, механизм правил обрабатывает сообщения из Главной темы и может при желании помещать их в другие темы с помощью узла правила «Контрольная точка». По умолчанию основная тема просто игнорирует неудачные сообщения. Это сделано для обратной совместимости с предыдущими выпусками. Однако вы можете перенастроить это на свой страх и риск. Обратите внимание: если одно сообщение не обрабатывается из-за сбоя в сценарии узла правил, это может помешать обработке следующих сообщений. Мы разработали специальную панель мониторинга для отслеживания обработки и сбоев Rule Engine.

Тема `HighPriority` может использоваться для доставки аварийных сигналов или других важных этапов обработки. Сообщения в теме `HighPriority` постоянно обрабатываются в случае сбоя, пока обработка сообщения не завершится успешно. Это полезно, если у вас отключился SMTP-сервер или внешняя система. Механизм правил будет повторять отправку сообщения, пока оно не будет обработано.

Тема `SequentialByOriginator` важна, если вы хотите убедиться, что сообщения обрабатываются в правильном порядке. Сообщения от одного и того же объекта будут обрабатываться в порядке поступления в очередь. Rule Engine не отправит новое сообщение в цепочку правил до тех пор, пока не будет подтверждено предыдущее сообщение для того же идентификатора объекта.

## 9. Библиотека встроенных узлов - примеры

### 9.1. Узлы фильтрации

Узлы фильтрации используются для фильтрации и маршрутизации сообщений.

#### 9.1.1. Check Relation Filter Node

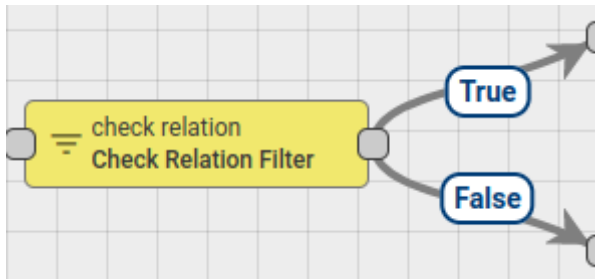


Рисунок 21.

Проверяет отношение выбранной сущности к отправителю сообщения по типу и направлению.

#### 9.1.2. Check Existence Fields Node

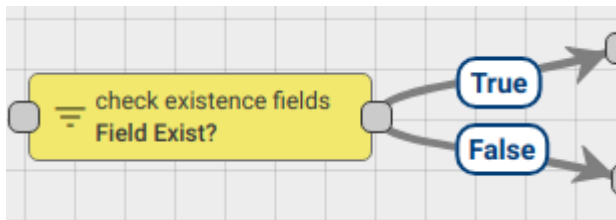


Рисунок 22.

Узел правила проверяет наличие полей данных и метаданных входящего сообщения.

#### 9.1.3. Message Type Filter Node

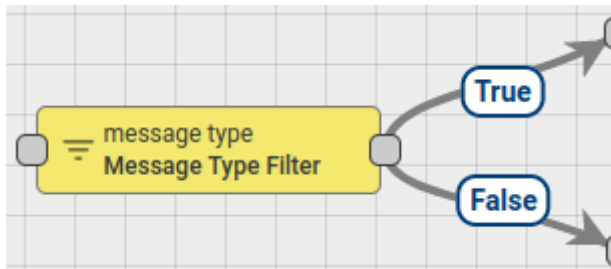


Рисунок 23.

В конфигурации узла определяется набор разрешенных типов сообщений для входящих сообщений. В системе есть predefined типы сообщений, такие как атрибуты сообщения, телеметрия сообщения, запрос RPC и т. Д. Также можно определить любые настраиваемые типы сообщений.

## 9.2. Узлы обогащения

Узлы обогащения используются для обновления метаданных входящего сообщения.

### 9.2.1. Calculate delta

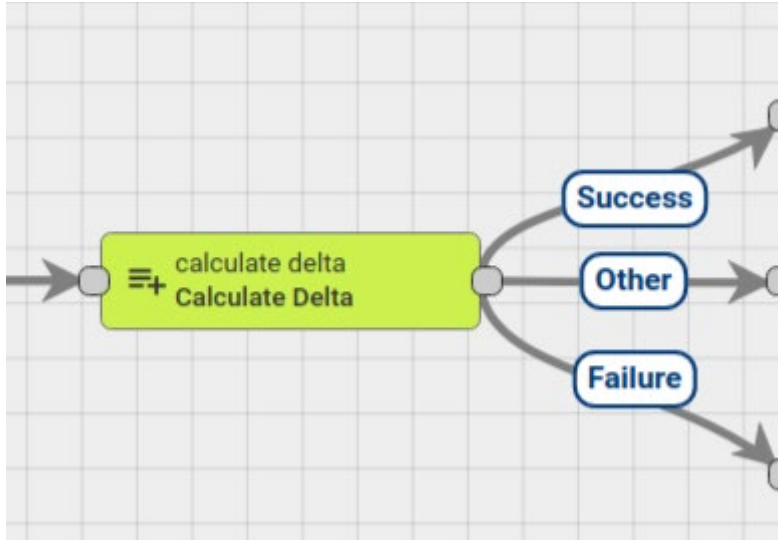


Рисунок 24.

Вычисляет «дельту» на основе предыдущего чтения временного ряда и текущего чтения и добавляет его к сообщению. Расчет дельты выполняется в рамках отправителя сообщения, например устройство, актив или клиент. Полезно для сценария использования интеллектуального измерения. Например, когда водомерный прибор сообщает абсолютное значение счетчика импульсов один раз в сутки. Чтобы узнать потребление за текущий день, вам нужно сравнить значение за предыдущий день со значением за текущий день.

Параметры конфигурации:

Ключ входного значения (по умолчанию «pulseCounter») - указывает ключ, который будет использоваться для вычисления дельты.

Ключ выходного значения (по умолчанию «дельта») - указывает ключ, который будет хранить значение дельты в расширенном сообщении.

Decimals - точность вычисления дельты.

Использовать кеш для последнего значения (по умолчанию «включено») - включает кеширование последних значений в памяти.

Сообщать «Failure», если дельта отрицательная (по умолчанию «разрешено») - принудительно завершает обработку сообщения, если значение дельты отрицательное.

Добавить период между сообщениями (по умолчанию «отключен») - добавляет значение периода между текущим и предыдущим сообщением.

Отношения узла правила:

Узел правила создает сообщение с одним из следующих отношений:

Успешно - если во входящем сообщении присутствует ключ, настроенный с помощью параметра «Ключ вводимого значения»;

Другое - если ключ, настроенный с помощью параметра «Ключ вводимого значения», отсутствует во входящем сообщении;

Отказ - если установлен параметр «Сообщать об отказе, если дельта отрицательная» и расчет дельты возвращает отрицательное значение;

### 9.2.2. Customer attributes

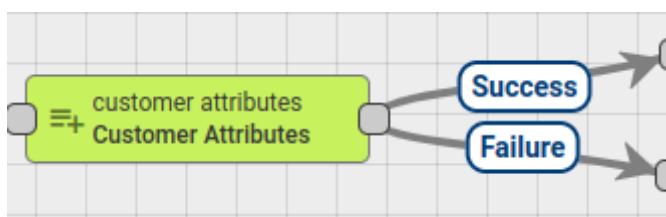


Рисунок 25.

Узел находит клиента объекта отправителя сообщения и добавляет атрибуты клиентов или значение последней телеметрии в метаданные сообщения.

Администратор может настроить соответствие между исходным именем атрибута и именем атрибута метаданных.

В конфигурации узла есть флажок «Последняя телеметрия». Если этот флажок установлен, узел будет получать последние данные телеметрии для настроенных ключей. В противном случае узел будет получать атрибуты области сервера.

## 9.3. Узлы трансформации

Узлы трансформации используются для изменения полей входящего сообщения, таких как отправитель, тип сообщения, полезная нагрузка и метаданные.

### 9.3.1. Change originator

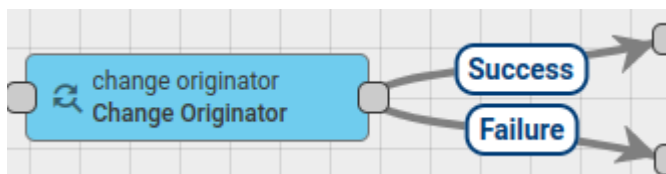


Рисунок 26.

Все входящие сообщения имеют поле отправителя, которое идентифицирует объект, отправляющий сообщение. Это может быть устройство, актив, клиент, арендатор и т. д.

Этот узел используется в случаях, когда отправленное сообщение должно обрабатываться как сообщение от другого объекта. Например, устройство отправляет

данные телеметрии, и данные телеметрии должны быть скопированы в актив более высокого уровня или клиенту. В этом случае администратор должен добавить этот узел перед узлом сохранения временных рядов.

«Создателя» можно изменить для:

- Заказчик отправителя;
- Арендатор отправителя;
- Связанная сущность, которая определяется запросом отношений.

В конфигурации «Запрос отношений» администратор может выбрать необходимое направление и уровень глубины отношения. Также набор фильтров отношений может быть настроен с требуемым типом отношения и типами сущностей.

### 9.3.2. Script Transformation Node

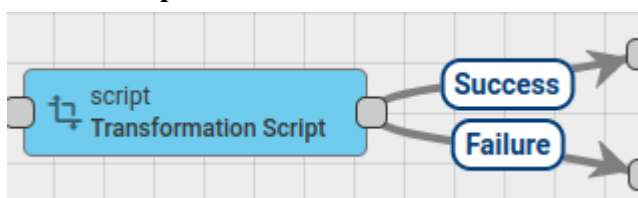


Рисунок 27.

Функция JavaScript получает 3 входных параметра:

1. msg - это полезная нагрузка сообщения.
2. метаданные - это метаданные сообщения.
3. msgType - это тип сообщения.

Скрипт должен вернуть следующую структуру:

```
{  
  
  msg: новая полезная нагрузка,  
  
  метаданные: новые метаданные,  
  
  msgType: новый msgType  
  
}
```

## 10. Практическая работа с цепочками правил

### 10.1. Вход в систему

Используйте интернет-браузер. На странице авторизации укажите логин и пароль, затем нажмите кнопку «Войти».

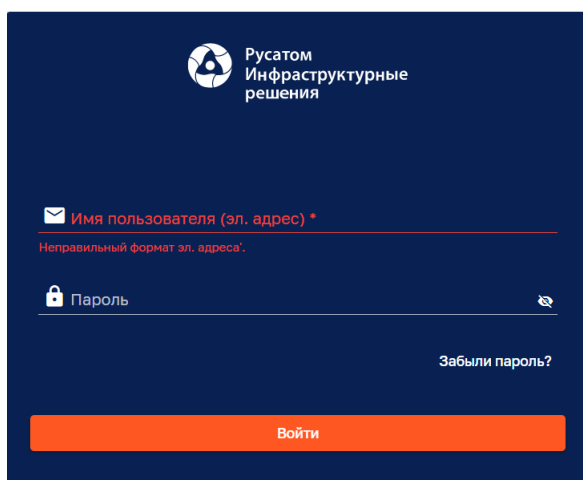


Рисунок 28. Вход в систему.

## 10.2. Главное меню

После входа в систему можно видеть Главное меню системы (см Рисунок 29):



Рисунок 29. Главное меню.

## 10.3. Панель «Цепочки правил»

Выберите на левой панели Главного меню опцию «Цепочки правил». Откроется страница со списком правил, имеющих в системе (см. Рисунок 30).

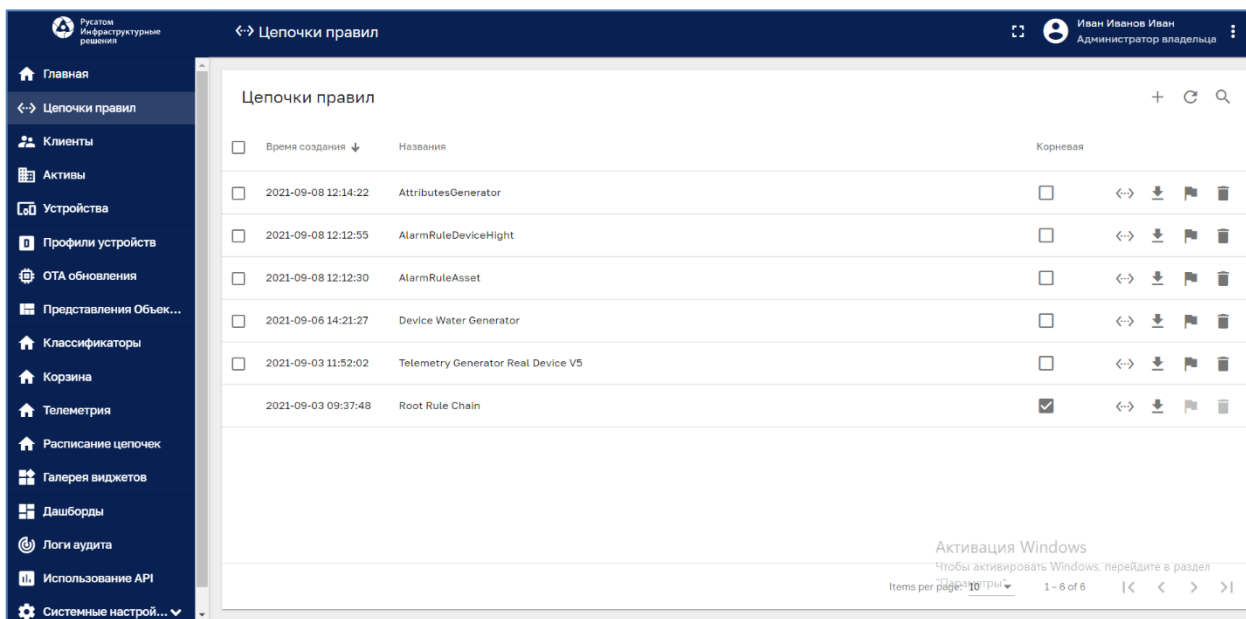


Рисунок 30. Цепочки правил.

#### 10.4. Просмотр цепочки

Для того, чтобы проанализировать структуру цепочки, нужно нажать на пиктограмму  в строке таблицы цепочек:

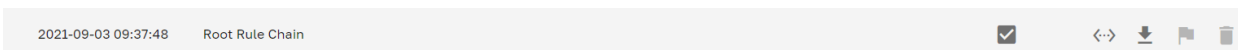



Рисунок 31.

#### 10.5. Экспорт цепочки в формате json

Для экспорта корневой цепочки нужно нажать на пиктограмму  в строке для Root Rule Chain.

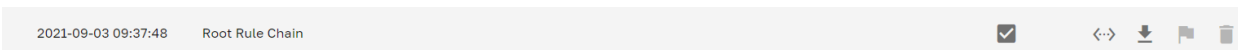


Рисунок 32.



## 10.6. Создание и редактирование цепочки

### 10.6.1. Создание новой цепочки

Рассмотрим вариант, когда новая цепочка создается на основе имеющегося экспортного json-файла.

Нажать на «+» на панели цепочек и выбрать опцию «Импортировать цепочку». Указать имя экспортного json-файла, затем нажать кнопку «Импортировать» (см. Рисунок 33):

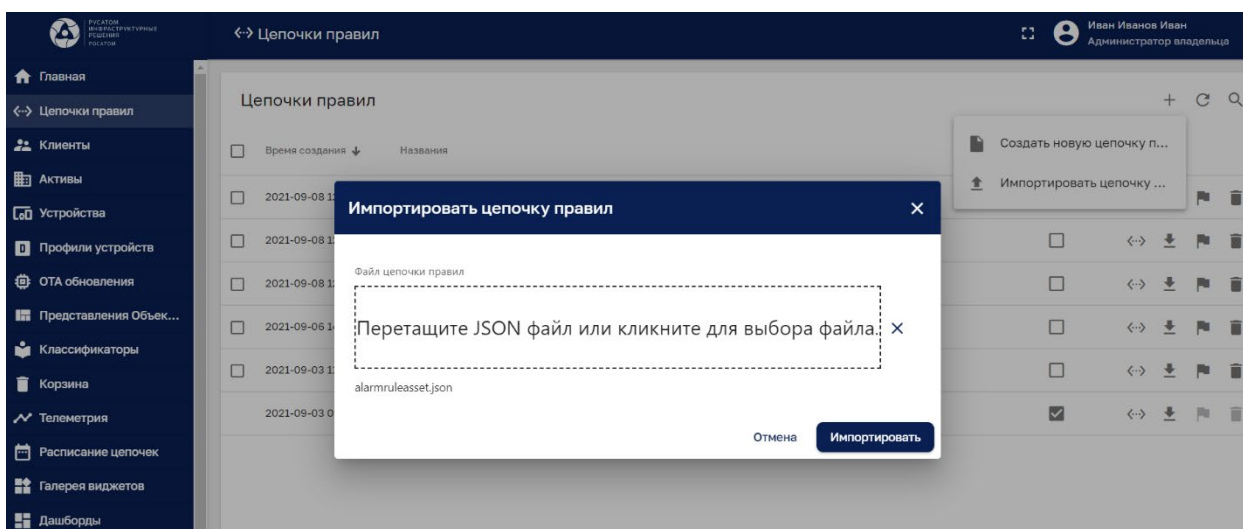



Рисунок 33. Создание новой цепочки путем импорта.

Будет создана копия цепочки, имя будет задублировано, и желательно произвести переименование.

### 10.6.2. Переименование новой цепочки

Для переименования цепочки надо щелкнуть по ней и на панели «Подробности о цепочке правил» нажать на кнопку редактирования , после чего можно скорректировать имя цепочки.

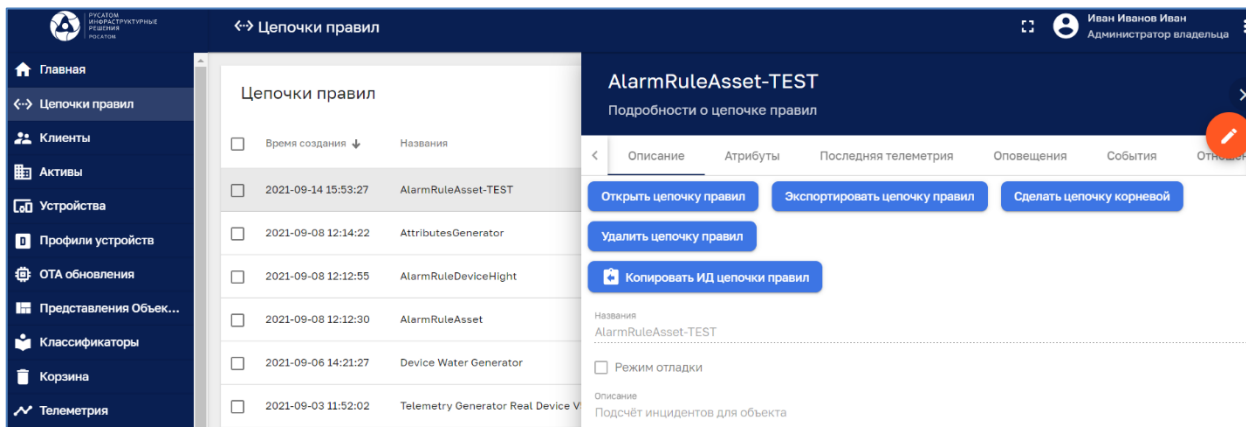


Рисунок 34.

### 10.6.3. Добавление узлов и связей

Добавляемый узел перетаскивается из второй вертикальной колонки правил на рабочее поле. Перетащим правило «script», далее система потребует ввести название:

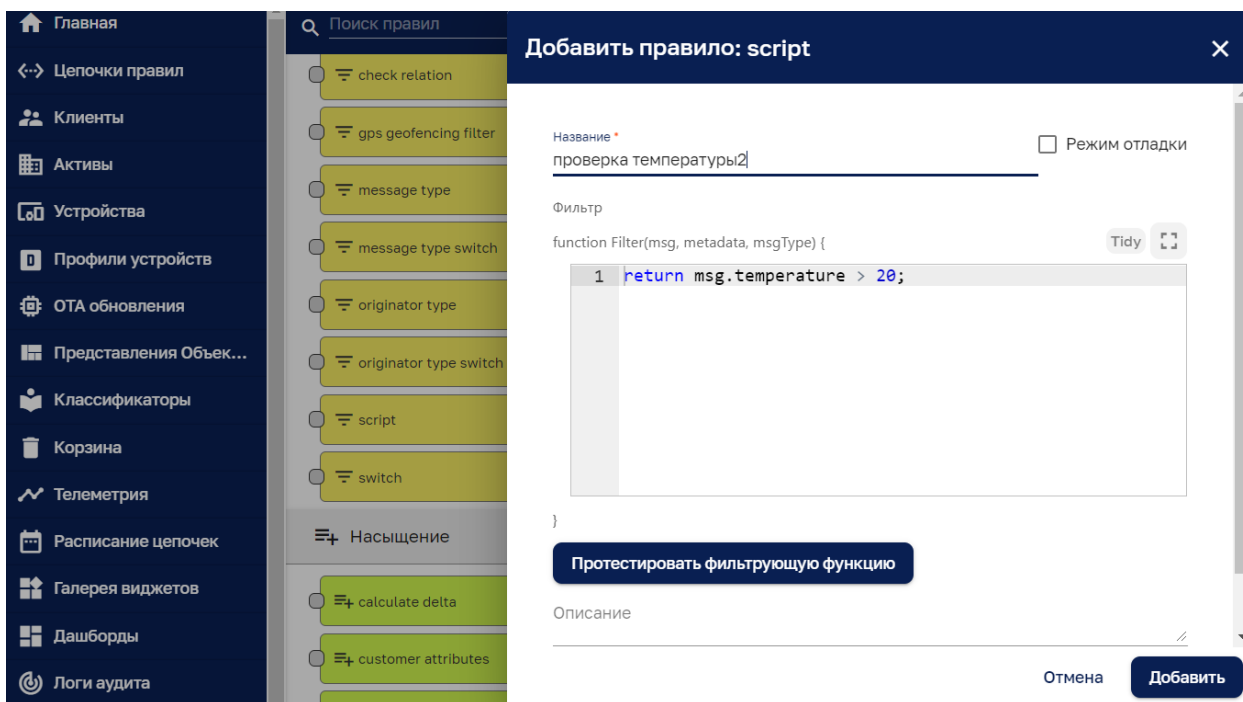


Рисунок 35.

Появился новый узел «проверка температуры 2».

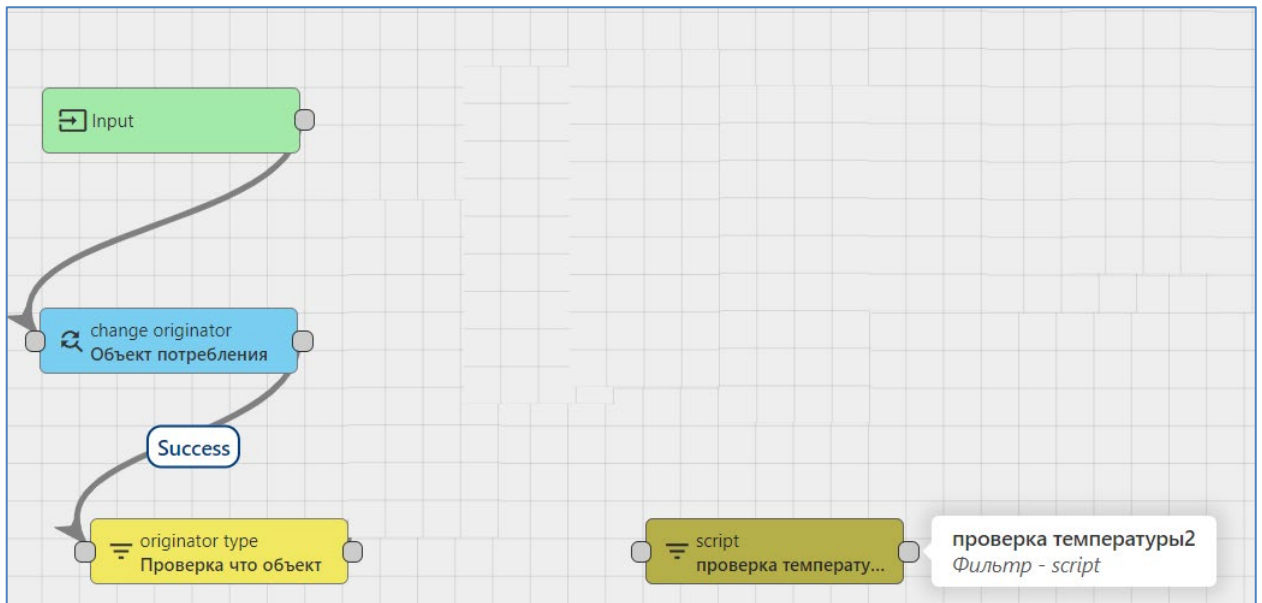


Рисунок 36.

Добавляем связи мышкой, щелкаем по стартовой точке на одном узле и тянем до финишной на другом узле. Система предлагает выбрать метку связи(например, True) из выпадающего меню. Затем нажимаем «Добавить»:

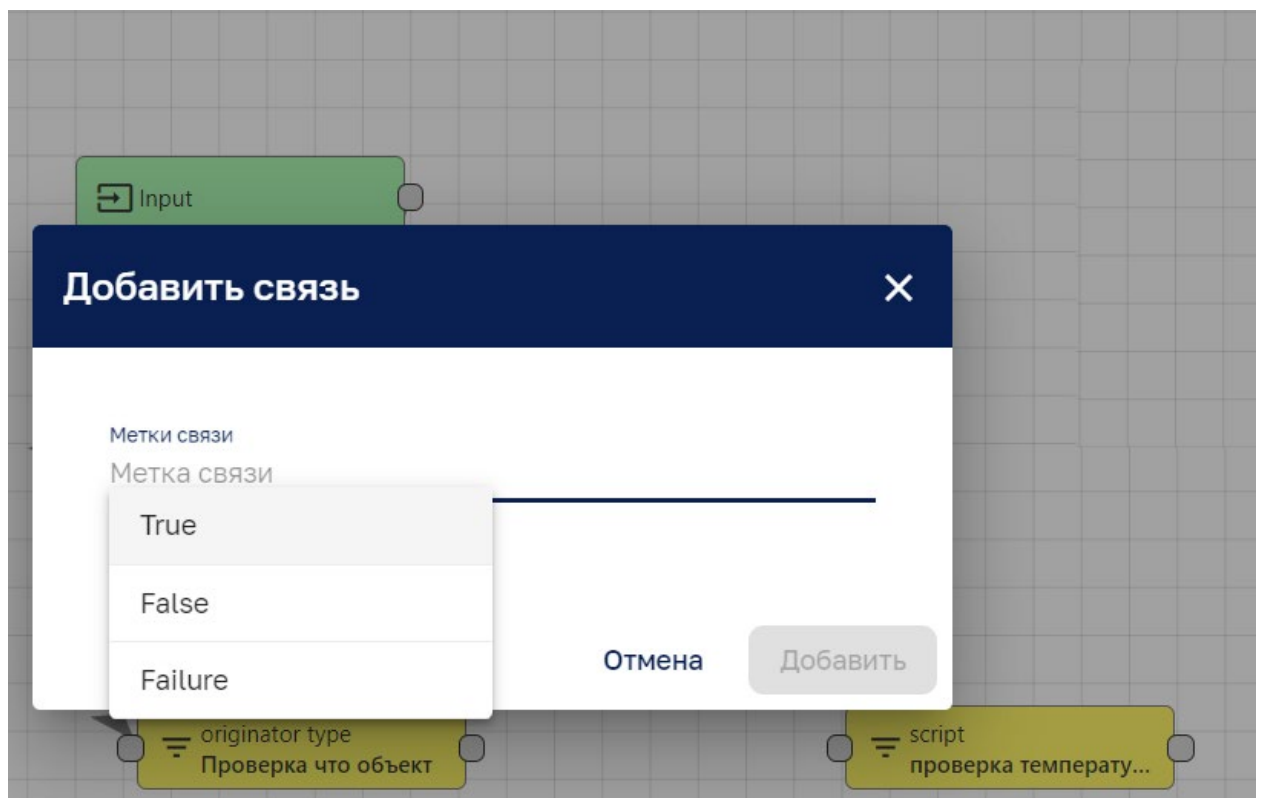


Рисунок 37.

Видно, что узел скрипта добавлен и имеет связь с узлом «originator type».

Аналогичным образом перетаскиваем на рабочее поле два новых узла «create alarm» и «clear alarm»:

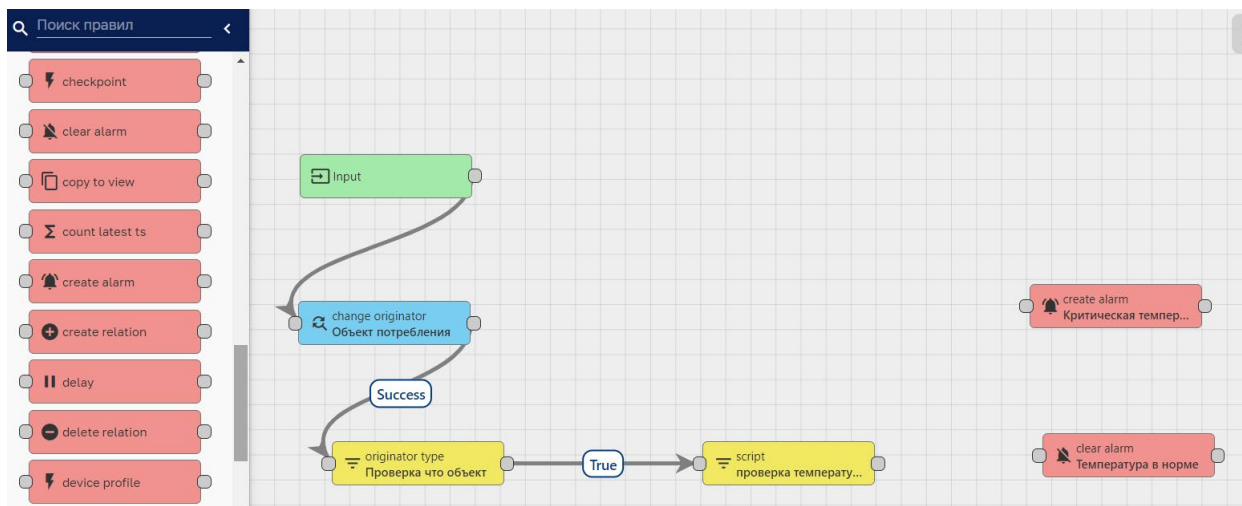


Рисунок 38.

По аналогии добавляем связь от скриптового узла к узлу «create alarm» с меткой «true»:

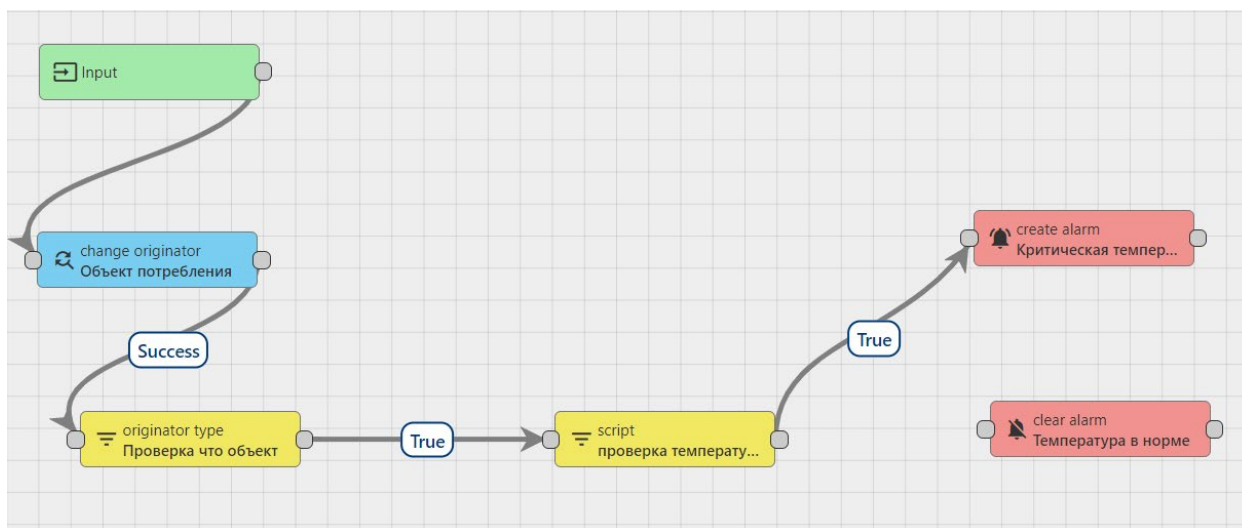


Рисунок 39.

По аналогии добавляем связь от скриптового узла к узлу «clear alarm» с меткой «false»:

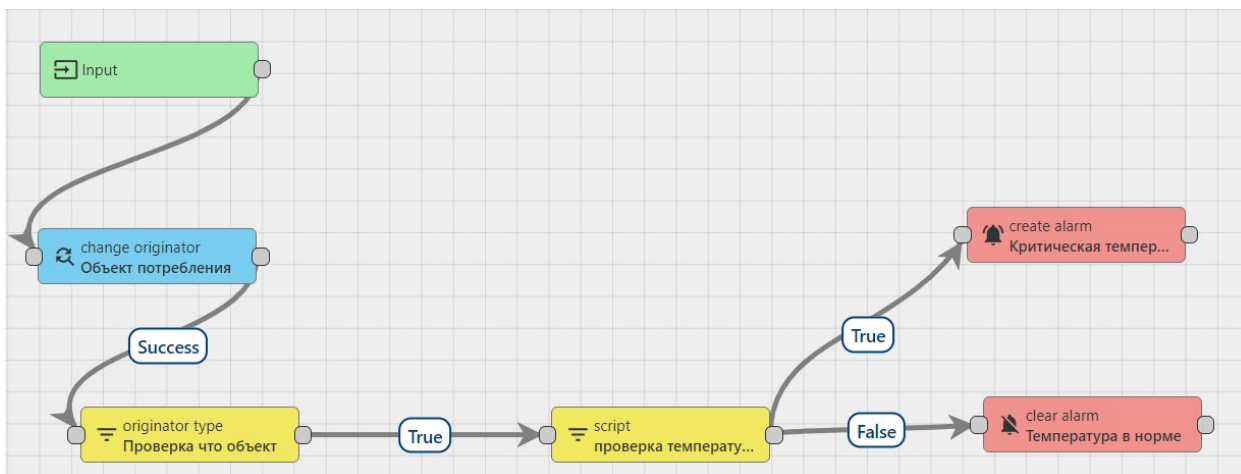
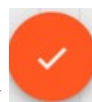


Рисунок 40.

### 10.6.4. Сохранение изменений



Для сохранения изменений необходимо нажать кнопку (см. Рисунок 41):

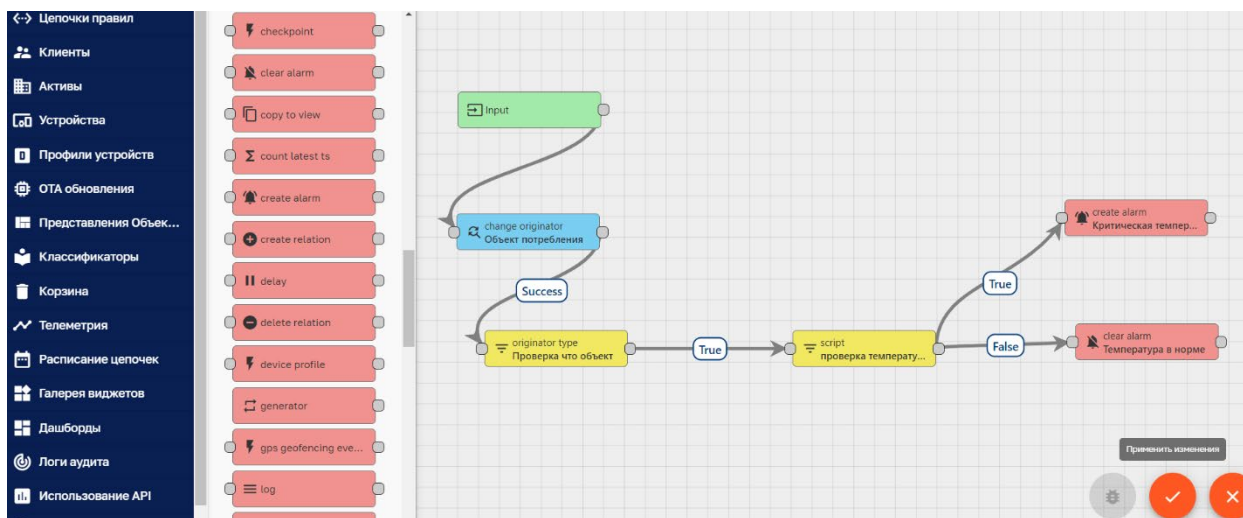


Рисунок 41. Сохранение изменений в цепочке.

### 10.6.5. Тестирование JS- скрипта

Некоторые узлы правил имеют особую функцию пользовательского интерфейса, которая позволяет тестировать функции JS.



Установите фокус на узел и нажмите:

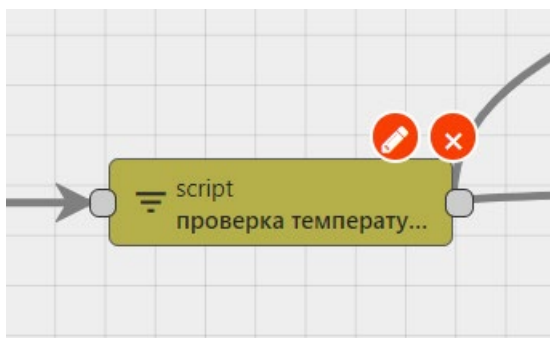


Рисунок 42.

Система покажет панель с телом JS-функции. Нажмите кнопку «Протестировать фильтрующую функцию»:

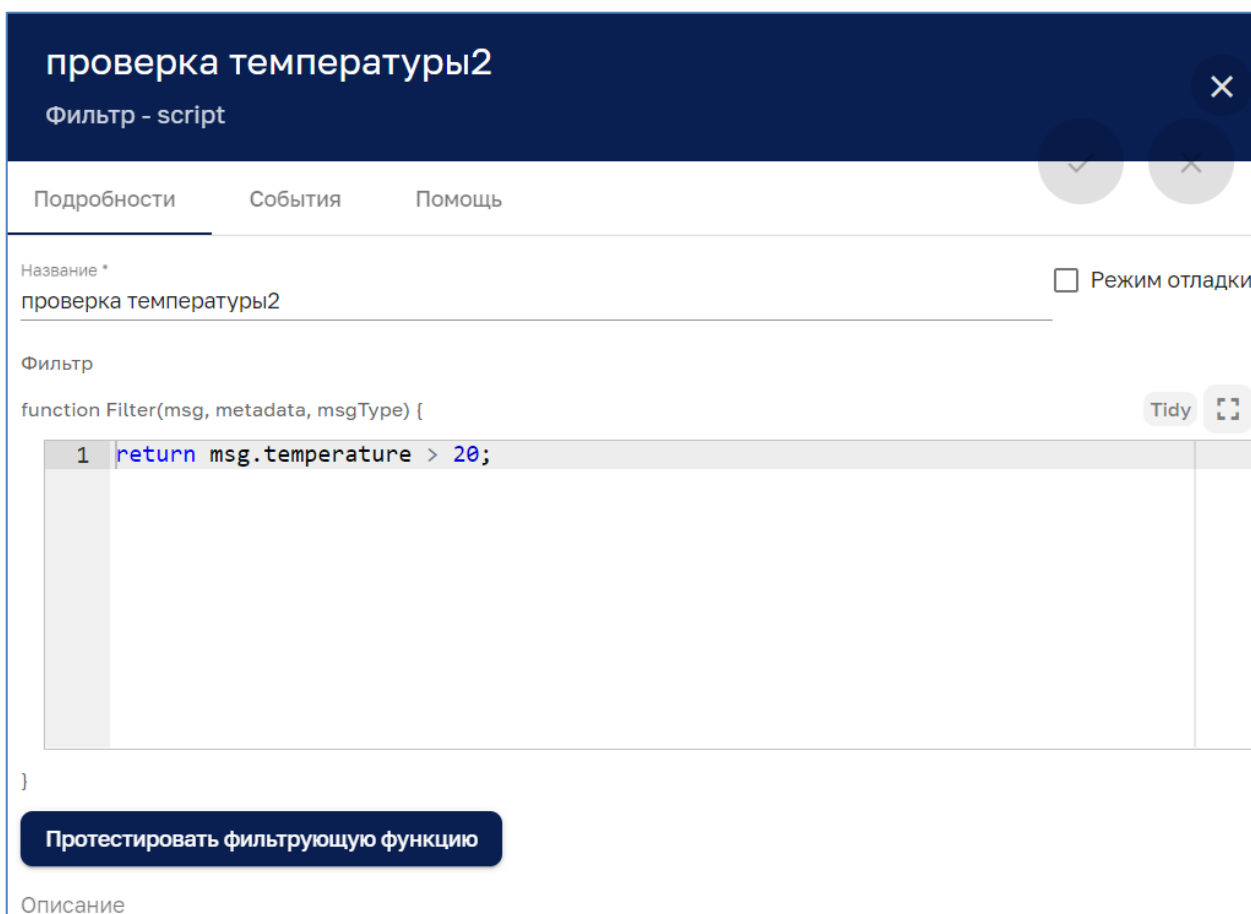


Рисунок 43. Панель JS

Редактор JS позволяет заменять входные параметры и проверять выходные данные функции.

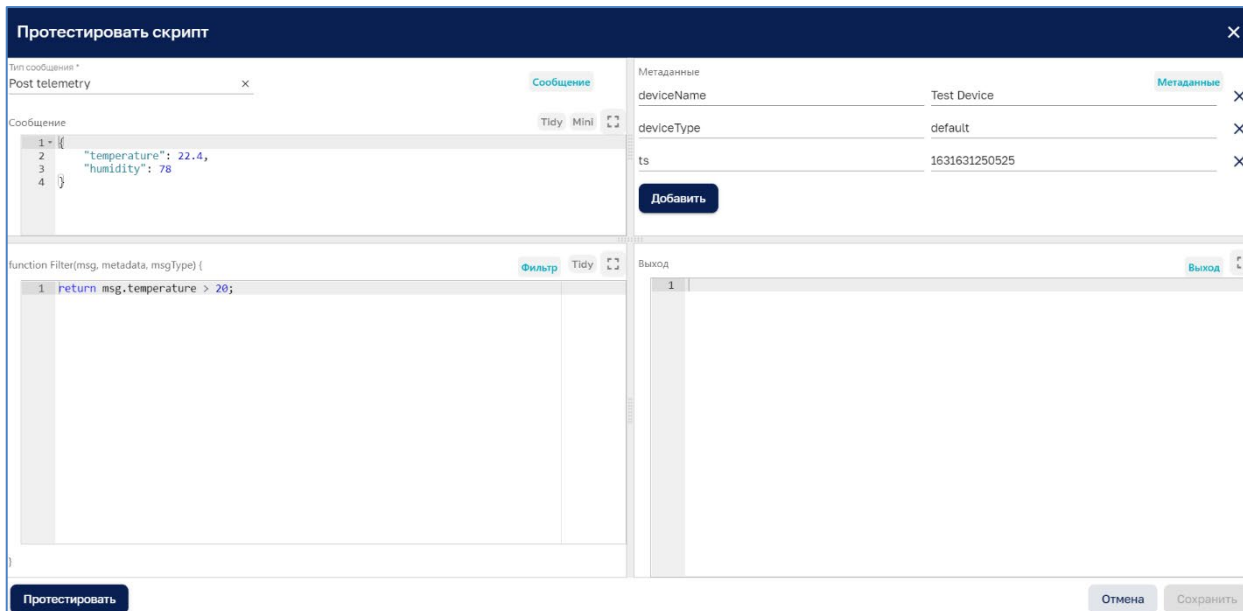


Рисунок 44. Панель редактора JS.

Можно определить:

- Тип сообщения в верхнем левом поле.
- Полезные данные сообщения в левом разделе сообщений.
- Метаданные в правом разделе Метаданные.
- Актуальный JS-скрипт в разделе «Фильтр».

После нажатия «Протестировать» вывод будет возвращен в правую секцию вывода.

## 11. Примеры решения прикладной задачи

### 11.1. Root Rule Chain

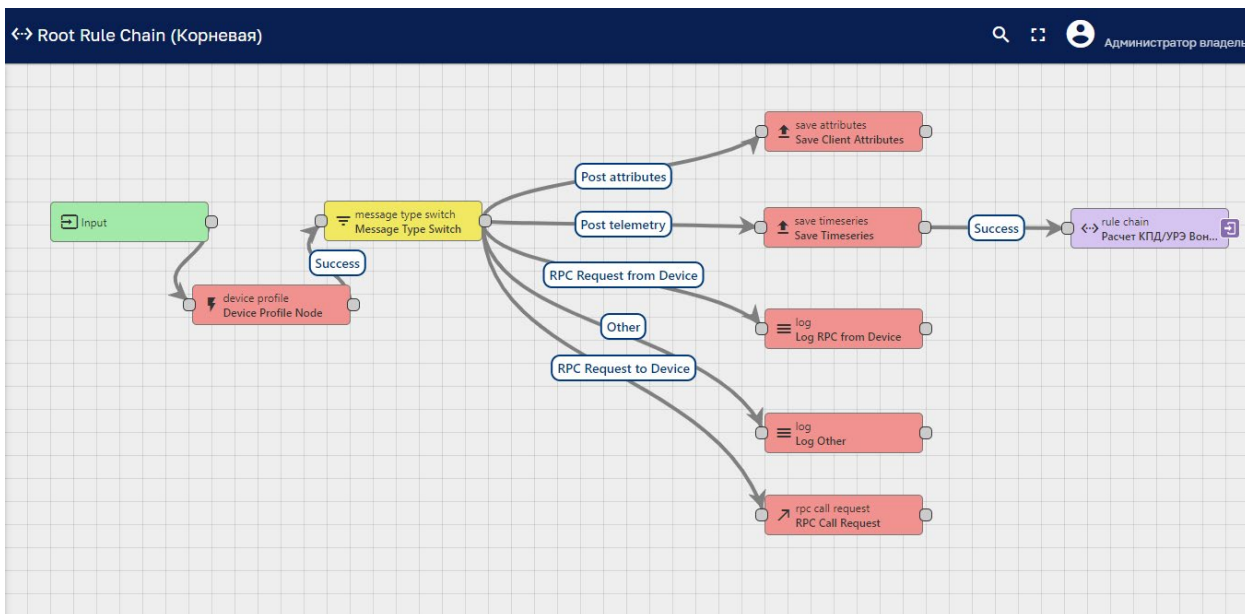


Рисунок 45. Пример Root Rule Chain

### 11.2. Расчет КПД\УРЭ Водонапорной станции

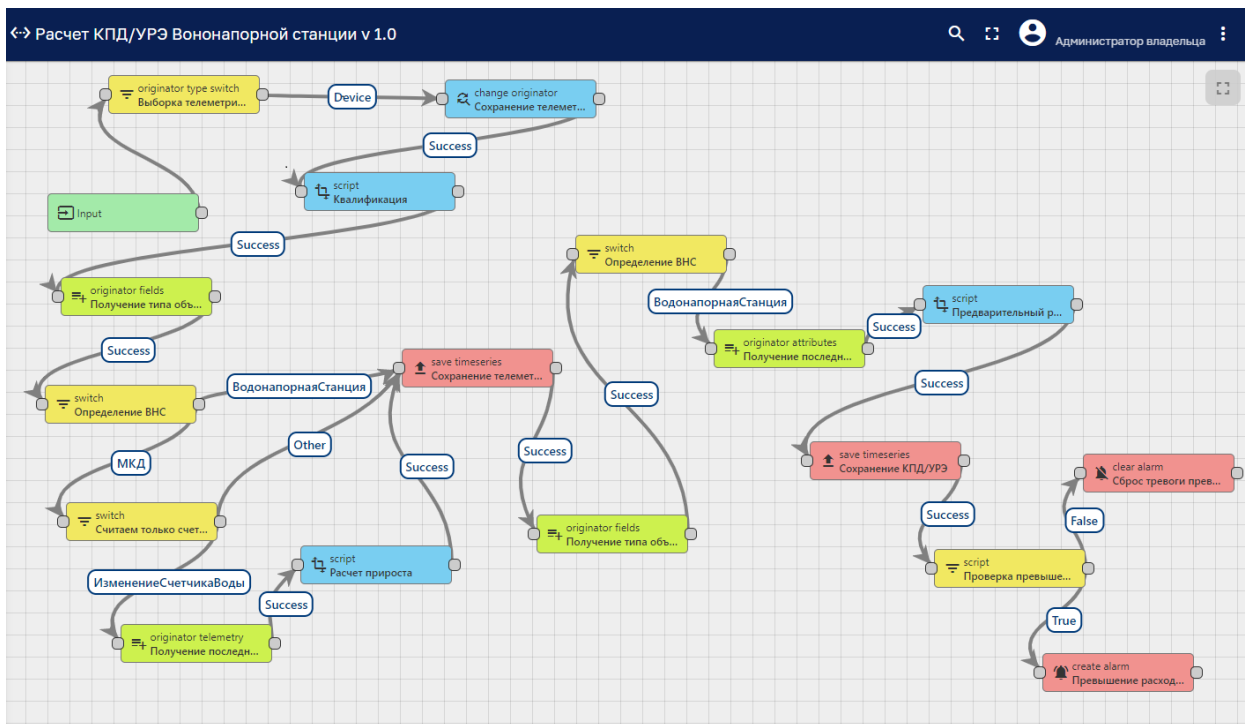


Рисунок 46. Пример решения прикладной задачи.



### 11.3. Расчет небалансовых потерь

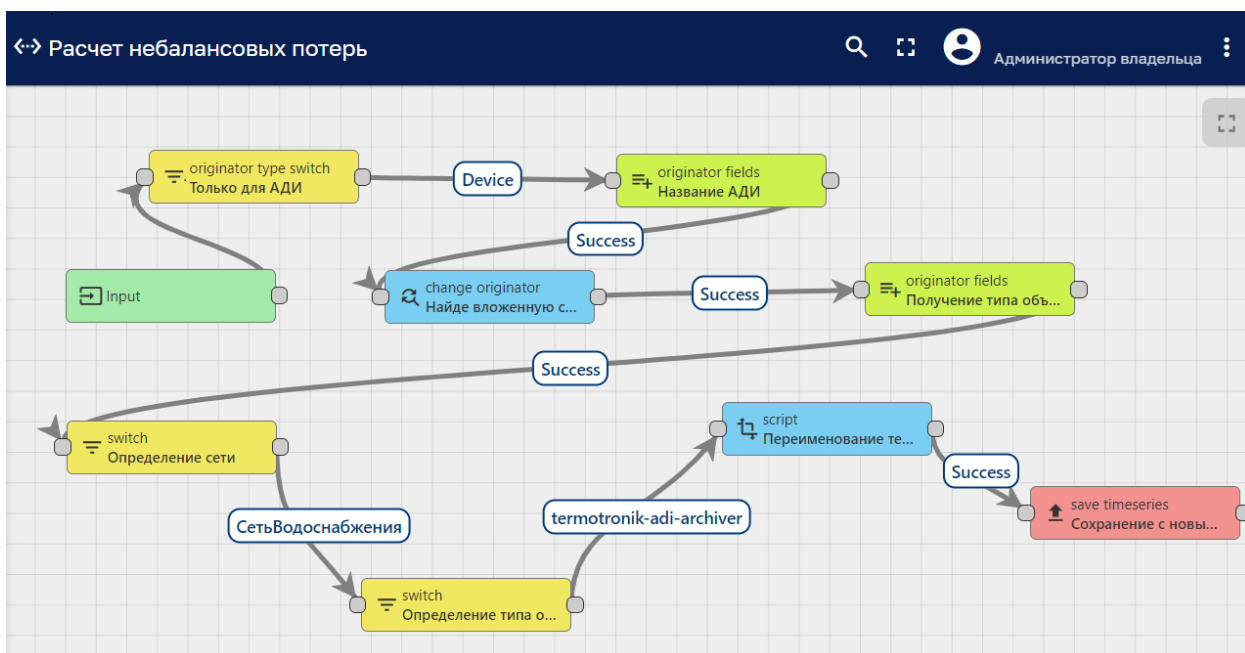


Рисунок 47. Пример расчета.

### 11.4. Расчет потери воды в сети водоснабжения

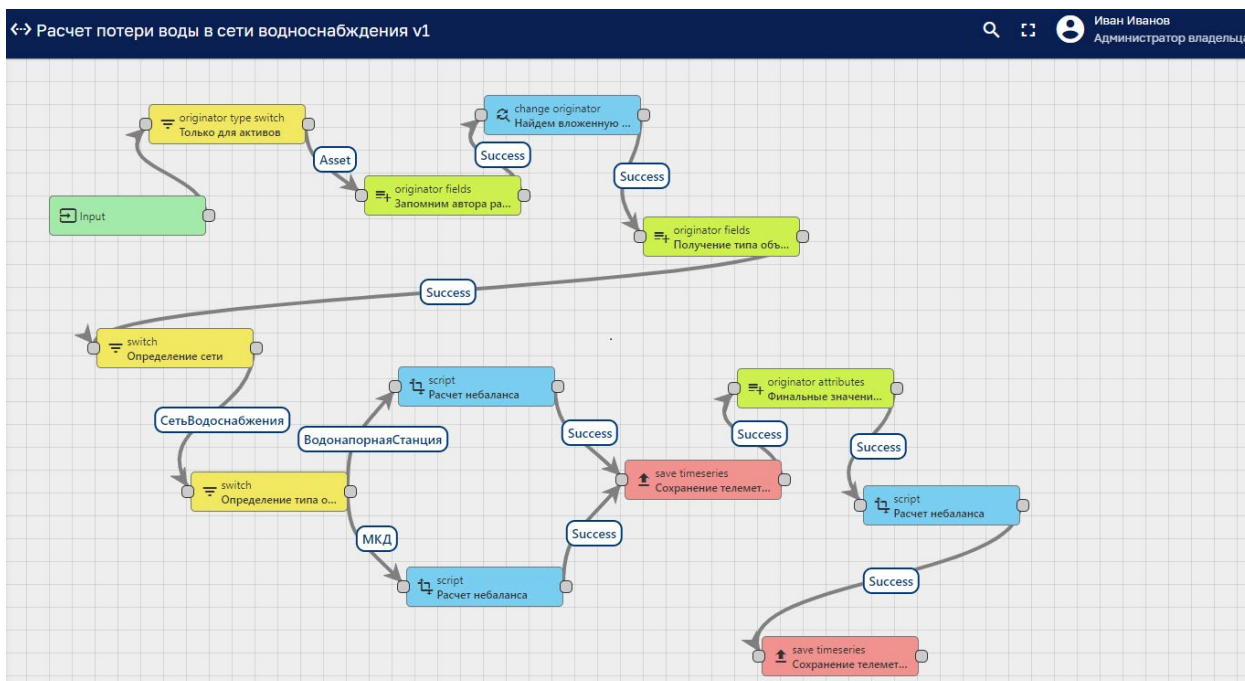


Рисунок 48. Пример расчета.

## 11.5. Генератор

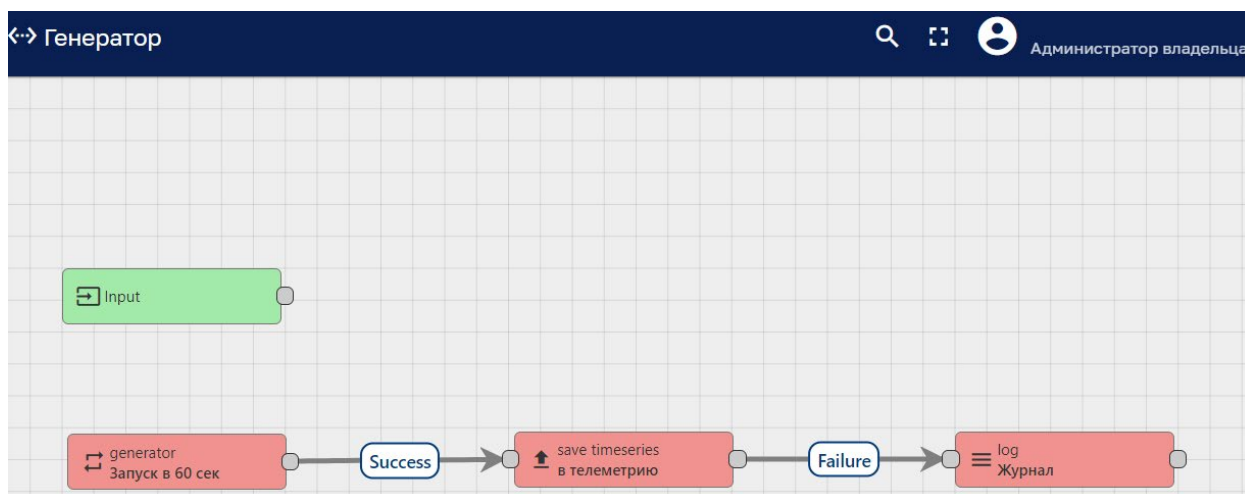


Рисунок 49. Пример использования «генератора».