Оглавление

[1. Описание приложения 2](#_Toc143781115)

[1.1 Общее 2](#_Toc143781116)

[1.2 Функции формул (делегаты) 2](#_Toc143781117)

[1.2.1 Работа с метками времени 2](#_Toc143781118)

[1.2.2 Операции с различными данными 5](#_Toc143781119)

[1.2.3 Операции с TSDB 6](#_Toc143781120)

[1.3 Правила формирования расчётных атрибутов 10](#_Toc143781121)

[1.4 Запуск по триггеру 13](#_Toc143781122)

[1.5 Пример создания расчёта 15](#_Toc143781123)

[1.5.1 Подготовка входных переменных 15](#_Toc143781124)

[1.5.2 Настройка меток времени 15](#_Toc143781125)

[1.5.3 Проверка на bad. Функция If в связке с BadVal 16](#_Toc143781126)

[1.5.4 Простые суммы и разности 16](#_Toc143781127)

[1.5.4 Расчёт с условиями 17](#_Toc143781128)

[1.5.4 Цикличные расчёты 17](#_Toc143781129)

[2. Запуск приложения на сервере 18](#_Toc143781130)

[2.1 Настройка файлов конфигурации 18](#_Toc143781133)

[2.1.1 appsettings.json 18](#_Toc143781134)

[2.1.2 NLog.config 19](#_Toc143781135)

[1.1.1 Nodes.json 19](#_Toc143781136)

[2.2 Настройка окружения 20](#_Toc143781137)

[3. Управление 23](#_Toc143781138)

[3.1 Добавление узла расчётов (перезапуск при наличии) 23](#_Toc143781139)

[3.2 Пересчёт узла расчётов 25](#_Toc143781140)

[3.3 Удаление узла расчётов 26](#_Toc143781141)

[1. Планируемые доработки функционала 27](#_Toc143781142)

[4.1 Вывод промежуточных результатов онлайн 27](#_Toc143781143)

[4.2 Реализовать прототип конфигуратора IFormula 27](#_Toc143781144)

[4.3 Реализовать при необходимости новые вычислительные функции 27](#_Toc143781145)

# Описание приложения

## Общее

Приложение разработано в целях упрощения конфигурации и запуска элементарных вычислений (по аналогии с AF Analises), для расчёта большого количества нешаблонных вычислений (индивидуально для каждого элемента модели).

Платформа приложения .NET 6.0 с использованием следующих внешних библиотек:

* NLog (осуществляет логирование);
* Quartz (планировщик задач);

Так же написаны следующие вспомогательные библиотеки на платформе .NET 6.0:

* ApplicationServices (бизнес логика работы с узлами расчётов, обработчики видов расчётов, модель планировщика, модель расчитываемого элемента и атрибута);
* Interpreter (интерпретатор синтаксиса формул- сам движок, вычисляющий формулы с учётом разработанных делегатов “функций”, работающий на внешней библиотеке DynamicExpresso, сами делегаты и модель их результатов);
* ISP.SDK (библиотека работы с MS SQL БД ISP, использующая System.Data.SqlClient, разработанная для получения конфигурации модели узлов расчётов IFormula);
* TSDBWorkerAPI (библиотека http клиент TSDB WebApi, использующая RestSharp и NewtonsoftJson, разработанная в целях чтения и записи настроек тегов, данных тегов, получения и чтения подписки обновления данных тегов. Так же содержит модель запросов и ответов.

## Функции формул (делегаты)

### 1.2.1 Работа с метками времени

#### 1.2.1.1 Bod (Получение метки времени начала суток)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. offset типа double (необязательный, по умолчанию = 0)

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением начала суток метки параметра dateTime и сдвигом в минутах = offset.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.2 Bom (Получение метки времени начала месяца)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. offset типа double (необязательный, по умолчанию = 0)

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением начала месяца метки параметра dateTime и сдвигом в минутах = offset.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.3 Boy (Получение метки времени начала года)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. offset типа double (необязательный, по умолчанию = 0)

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением начала года метки параметра dateTime и сдвигом в минутах = offset.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.4 DaysInMonth (Получение количества дней в месяце)

Функция на входе имеет 2 параметра:

1. year типа int;
2. month типа int

Функция вернёт объект типа int со значением количества дней в месяце параметра month года параметра year.

#### 1.2.1.5 AddDays (Изменение метки времени на количество дней)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. count типа double

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением параметра dateTime и добавленным количеством дней = count.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.6 AddHours (Изменение метки времени на количество часов)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. count типа double

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением параметра dateTime и добавленным количеством часов = count.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.7 AddMinutes (Изменение метки времени на количество минут)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. count типа double

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением параметра dateTime и добавленным количеством минут = count.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.8 AddSeconds (Изменение метки времени на количество секунд)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. count типа double

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением параметра dateTime и добавленным количеством секунд = count.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.9 AddMonths (Изменение метки времени на количество месяцев)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. count типа double

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением параметра dateTime и добавленным количеством месяцев = count.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.10 AddYears (Изменение метки времени на количество лет)

Функция на входе имеет 2 параметра:

1. dateTime типа object;
2. count типа double

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа DateTime со значением параметра dateTime и добавленным количеством лет = count.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.11 Year (Получение значения года)

Функция на входе имеет 1 параметр:

1. dateTime типа object

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа int со значением года метки времени параметра dateTime.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.12 Month (Получение значения месяца)

Функция на входе имеет 1 параметр:

1. dateTime типа object

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа int со значением месяца метки времени параметра dateTime.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.13 Day (Получение значения дня)

Функция на входе имеет 1 параметр:

1. dateTime типа object

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа int со значением дня метки времени параметра dateTime.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.14 Hour (Получение значения часа)

Функция на входе имеет 1 параметр:

1. dateTime типа object

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа int со значением часа метки времени параметра dateTime.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.15 Minute (Получение значения минут)

Функция на входе имеет 1 параметр:

1. dateTime типа object

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа int со значением минут метки времени параметра dateTime.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.16 Second (Получение значения секунд)

Функция на входе имеет 1 параметр:

1. dateTime типа object

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа int со значением минут метки времени параметра dateTime.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.1.17 DaySec (Получение количества секунд с начала суток)

Функция на входе имеет 1 параметр:

1. dateTime типа object

Функция при успехе преобразования dateTime параметра в тип DateTime вернёт объект типа double со значением секунд от начала суток метки времени параметра dateTime до метки времени параметра dateTime.

В случае ошибки преобразования dateTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

### 1.2.2 Операции с различными данными

#### 1.2.2.1 IF (условный оператор проверки bool результата выражения и возврат соответствующих значений)

Функция на входе имеет 3 параметра:

1. condition типа bool;
2. trueCondition типа object;
3. falseCondition типа object

Функция вернёт значения параметра trueCondition если значение параметра condition = true, иначе функция вернёт falseCondition типа object.

#### 1.2.2.2 Floor (округление вещественного числа до целого)

Функция на входе имеет 1 параметр:

1. input типа object

Функция при успехе преобразования input параметра в тип Decimal вернёт объект типа int со значением округления до целого числа параметра input.

В случае ошибки преобразования input в тип Decimal функция выдаст соответствующую ошибку.

#### 1.2.2.3 Compare (сравнение строк, не зависимо от регистра)

Функция на входе имеет 2 параметра:

1. str1 типа object;
2. str2 типа object

Функция сравнивает значения параметров str1 и str2 приведенных к типу string методом ToString() и к нижнему регистру (не зависит от регистра). Если строки равны, функция выдаст значения true типа bool, иначе false.

#### 1.2.2.4 Left (вырезание левой части строки до указанной длины)

Функция на входе имеет 2 параметра:

1. input типа object;
2. len типа int

Функция обрезает значение параметра input приведенного к типу string методом ToString() с 0 индекса до значения параметра len и возвращает полученное значение типа string .

### 1.2.3 Операции с TSDB

#### 1.2.3.1 Описание класса TSDBResult

Класс имеет следующие свойства:

* Value типа object (по умолчанию = “NoData”);
* Time типа DateTime (по умолчанию = DateTime.MinValue);
* Digital типа string (по умолчанию = “NoData”);
* Good типа bool (по умолчанию = false).

Метод ToString() имеет переопределение и возвращает значений метода ToString() свойства Value.

При применении арифметических операторов (сложение, вычитание, деление, умножение, >= или <=) к объекту данного класса операции будут производиться со свойством Value, преобразованными к типу double.

#### 1.2.3.2 TagVal (получение архивного значения тега, объекта типа TSDBResult на указанную метку времени)

Функция на входе имеет 3 параметра:

1. tagName типа string;
2. timestamp типа object;
3. method типа string (не обязательный, по умолчанию = "ExactOrPrev").

Перечень доступных значений параметра method (регистр не важен):

* exactorprev (точно на метку или ближайшее предыдущее);
* exactornext (точно на метку или ближайшее будущее);
* next (ближайшее будущее);
* prev (ближайшее предыдущее).

Функция при успехе преобразования timestamp параметра в тип DateTime возвращает объект типа TSDBResult (архивное значение) тега tagName на метку времени timestamp методом method.

В случае пустого значения параметра tagName функция выдаст ошибку "Неверный формат даты".

В случае значения параметра method не входящих в перечень досутпных методов поиска значения функция выдаст ошибку "Не распознан метод получения значения".

В случае ошибки преобразования параметра timestamp в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.3 Описание класса TAGInfo

Класс имеет следующие свойства:

* Desc типа string;
* EU типа string;
* Span типа int;
* Zero типа int;
* InstrumentTag типа string.

#### 1.2.3.4 TagInfo (получение атрибутов тега)

Функция на входе имеет 1 параметр:

1. tagName типа string

Функция возвращает объект типа TAGInfo тега tagName.

#### 1.2.3.5 BadVal (проверка значения на Bad качество)

Функция на входе имеет 1 параметр:

1. value типа TSDBResult

Функция возвращает объект типа bool со значением = true, если у объекта value свойство Good = false, иначе вернёт false.

#### 1.2.3.6 TagTot (получение суммарного значения тега за указанный интервал времени)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. calculationBasis типа string (не обязательный, по умолчанию = "time");
5. factor типа int (не обязательный, по умолчанию = 24).

Перечень доступных значений параметра calculationBasis (регистр не важен):

* time (интеграл по времени);
* event (интеграл по событию).

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- суммарное архивных значений тега tagName за интервал с метки времени параметра startTime по метку времени параметра endTime методом calculationBasis и умноженное на значение factor.

В случае значения параметра calculationBasis не входящих в перечень доступных значения функция примет его за “time”.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.7 TagAvg (получение среднего значения тега за указанный интервал времени)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. calculationBasis типа string (не обязательный, по умолчанию = "event").

Перечень доступных значений параметра calculationBasis (регистр не важен):

* time (интеграл по времени);
* event (интеграл по событию).

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- среднее архивных значений тега tagName за интервал с метки времени параметра startTime по метку времени параметра endTime методом calculationBasis.

В случае значения параметра calculationBasis не входящих в перечень доступных значения функция примет его за “event”.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.8 TagMax (получение максимального значения тега за указанный интервал времени)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. calculationBasis типа string (не обязательный, по умолчанию = "event").

Перечень доступных значений параметра calculationBasis (регистр не важен):

* time (интеграл по времени);
* event (интеграл по событию).

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- максимальное архивных значений тега tagName за интервал с метки времени параметра startTime по метку времени параметра endTime методом calculationBasis.

В случае значения параметра calculationBasis не входящих в перечень доступных значения функция примет его за “event”.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.9 TagMin (получение минимального значения тега за указанный интервал времени)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. calculationBasis типа string (не обязательный, по умолчанию = "event").

Перечень доступных значений параметра calculationBasis (регистр не важен):

* time (интеграл по времени);
* event (интеграл по событию).

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- минимальное архивных значений тега tagName за интервал с метки времени параметра startTime по метку времени параметра endTime методом calculationBasis.

В случае значения параметра calculationBasis не входящих в перечень доступных значения функция примет его за “event”.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.10 TagCount (получение количества значений тега за указанный интервал времени)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. calculationBasis типа string (не обязательный, по умолчанию = "event").

Перечень доступных значений параметра calculationBasis (регистр не важен):

* time (интеграл по времени);
* event (интеграл по событию).

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа int- количество архивных значений тега tagName за интервал с метки времени параметра startTime по метку времени параметра endTime методом calculationBasis.

В случае значения параметра calculationBasis не входящих в перечень доступных значения функция примет его за “event”.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.11 TimeEq (получение количества секунд, когда значение тега за указанный интервал времени равнялось указанному)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. value типа object.

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- количество секунд за интервал с метки времени параметра startTime по метку времени параметра endTime, когда значение тега равнялось значению параметра value.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.12 TimeGE (получение количества секунд, когда значение тега за указанный интервал времени было больше или равно указанному)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. value типа object.

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- количество секунд за интервал с метки времени параметра startTime по метку времени параметра endTime, когда значение тега было больше или равно значению параметра value.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.13 TimeGT (получение количества секунд, когда значение тега за указанный интервал времени было больше указанного)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. value типа object.

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- количество секунд за интервал с метки времени параметра startTime по метку времени параметра endTime, когда значение тега было больше значения параметра value.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.14 TimeLE (получение количества секунд, когда значение тега за указанный интервал времени было меньше или равно указанному)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. value типа object.

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- количество секунд за интервал с метки времени параметра startTime по метку времени параметра endTime, когда значение тега было меньше или равно значению параметра value.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.15 TimeLT (получение количества секунд, когда значение тега за указанный интервал времени было меньше указанного)

Функция на входе имеет 5 параметров:

1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. value типа object.

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- количество секунд за интервал с метки времени параметра startTime по метку времени параметра endTime, когда значение тега было меньше значения параметра value.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

#### 1.2.3.16 TimeNE (получение количества секунд, когда значение тега за указанный интервал времени не равнялось указанному)

Функция на входе имеет 5 параметров:

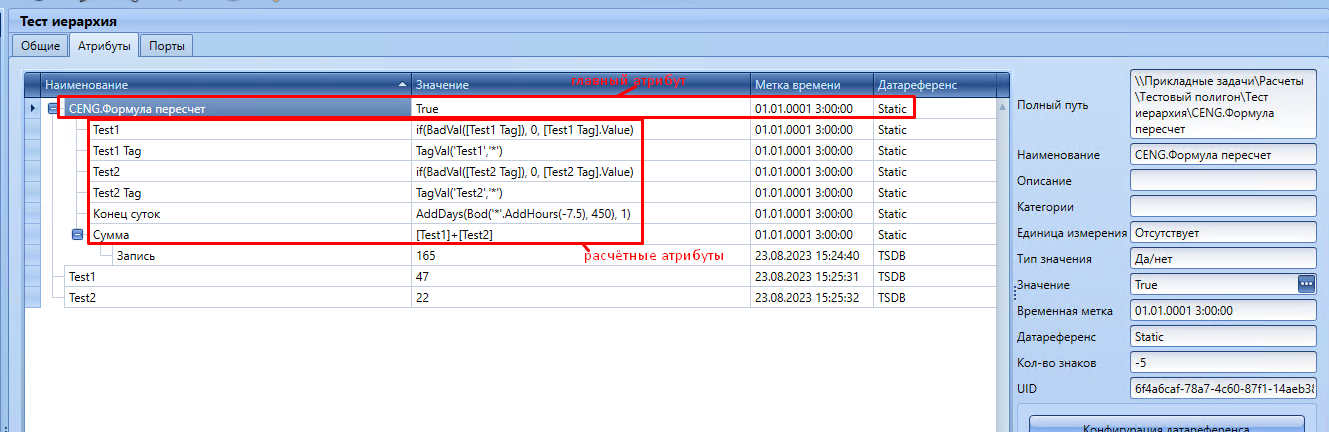
1. tagName типа string;
2. startTime типа object;
3. endTime типа object;
4. value типа object.

Функция при успехе преобразования startTime и endTime параметров в тип DateTime возвращает объект типа double- количество секунд за интервал с метки времени параметра startTime по метку времени параметра endTime, когда значение тега не равнялось значению параметра value.

В случае ошибки преобразования параметров startTime и endTime в тип DateTime функция выдаст ошибку "Неверный формат даты".

## Правила формирования расчётных атрибутов

Расчётные атрибуты являются дочерними Static атрибута типа Да/нет с названием узла расчёта из конфигурационного файла Nodes.json и значением True:



Расчётные атрибуты должны быть Static типа Текст. В значение прописываются формулы (можно использовать стандартные операторы C#, а так же описанные в п.1.2 функции).

Обращение к тегам внешних атрибутов (те, которые не расчётные) происходит через одинарные кавычки ‘Имя атрибута’:

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Обращение к результатам вычислений расчётных атрибутов происходит через квадратные скобки - [Имя расчётного атрибута]:

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Подставить в функцию текущую метку времени можно с помощью подстановки следующей записи ‘\*’:

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, число

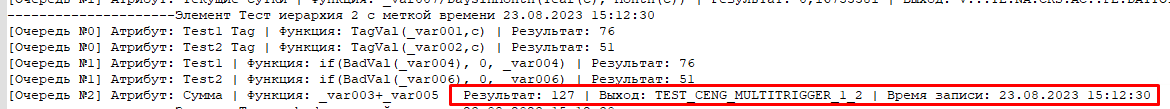
Автоматически созданное описание

Для записи результата вычисления расчётного атрибута в тег необходимо создать вложенный в расчётный атрибут с источником тега, в который мы собираемся записывать результат и назвать его «Запись» (строгое наименование):

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, веб-страница

Автоматически созданное описание

Соответствующая запись отразится в логе расчёта:



Для записи на конкретную метку времени необходимо создать вложенный в расчётный атрибут static атрибут типа Текст со значением конкретной метки времени или ссылке на расчётный атрибут, содержащий метку времени и именем «Время» (строгое наименование):

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Соответствующая запись будет в логе расчёта:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

## 1.4 Запуск по триггеру

Для запуска по триггеру в элементе необходимо добавить вложенный static атрибут «Триггер» (строгое наименование) типа текст (внутри искомого атрибута SearchAttribute из Nodes.json), в значение через запятую необходимо прописать имя внешнего атрибута с тегом TSDB, по приходу значения которого будет запускаться расчёт элемента с соответствующей меткой времени:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

При запуске узла расчётов с таким элементом будет указано количество элементов, запускаемых по триггеру и на соответствующие теги будет оформлена подписка на обновление значений в TSDB WebApi:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

При приходе значения в логах планировщика в каталоге TimersAndTriggers\Trigger будет запись нового значения триггерного тега, на метку которого запускается расчёт соответствующего элемента:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, программное обеспечение, веб-страница, Веб-сайт

Автоматически созданное описание

При запуске по триггеру (либо при неуспешной проверке подписки) получается ID подписки на соответтсвующие теги с временем жизни подписки = 10 часов, максимальным количеством точек в буффере = 200 \* количество тегов в подписке запросом /Subscription методом Post (п.1.10.1 “Руководство программиста I-DS-TSDB”).

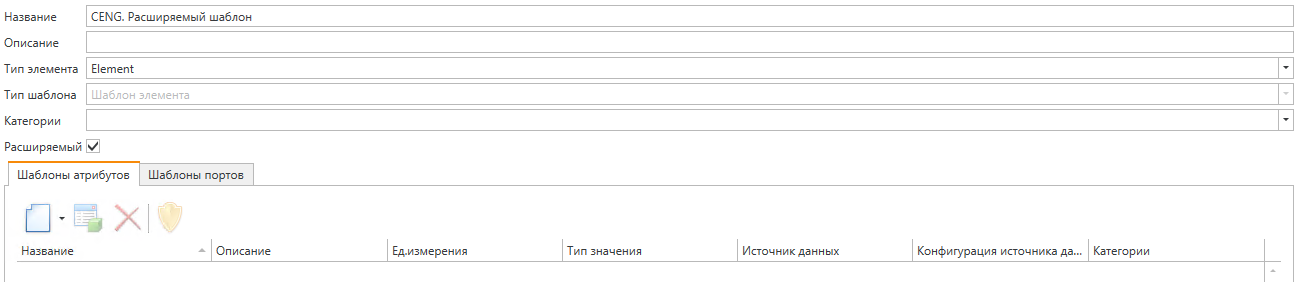
После инициализации узла расчёта по триггеру каждые 5 секунд сначала проверяется наличие подписки запросом /Subscription/{SubscriptionGuid} метод Get (п.1.10.4 “Руководство программиста I-DS-TSDB”), затем посылается запрос /Subscription/Data метод Get для получения данных из буфера подписки (п.1.10.5 “Руководство программиста I-DS-TSDB”).

В случае окончания времени жизни подписки, время её жизни продлевается запросом /Subscription/{SubscriptionGuid} метод Put (п.1.10.2 “Руководство программиста I-DS-TSDB”).

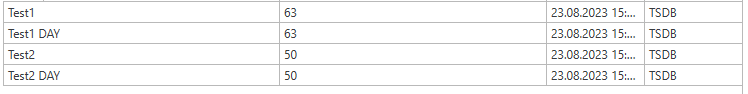
## 1.5 Пример создания расчёта

## 1.5.1 Подготовка входных переменных

Предварительно для элементов с расчётами необходимо создать расширяемый шаблон



Создав элемент по шаблону вводим входные переменные через атрибут с датареференсом TSDB



- Создаём атрибут с именем узла (например, CENG.Формула) с типом значения Да/Нет и датареференсом Static. Выбор значения Да/Нет определяет наличие расчёта в узле.



- Создаём вложенный в CENG.Формула атрибут для входной переменной в следующей форме:



TagVal(**'Имя атрибута'**,**'Метка времени'**)

TagVal – функция получения значений из тега

**'Имя атрибута'** – имя внешнего атрибута, задаётся через одинарные кавычки

**'Метка времени'** – определяет по какой метке времени берётся значение (по умолчанию '\*' – актуальная метка из архива)

Опционально можно задать настраиваемую метку путём замены 'Метка времени' на ссылку к вложенному в CENG.Формула атрибуту через квадратные скобки. Например TagVal('Имя атрибута',[Конец суток])



## 1.5.2 Настройка меток времени

Основные функции:

Bod('\*') – начало суток;

Bom('\*') – начало месяца;

Boy('\*') – начало года.

На большинстве производственных площадок отсчёт коммерческих суток ведётся со сдвигом относительно 00:00 часов.

Для задачи сдвига применяются следующие функции:

AddMinutes – минуты;

AddHours – часы;

AddDays – дни;

AddMonths – месяцы;

AddYears – годы.

Настройка сдвига:  
Bod('\*'.**AddHours(-7.5), 450**) – Начало суток 07:30;

Bom('\*'.**AddHours(-7.5), 450**) – Начало месяца 07:30;

Boy('\*'.**AddHours(-7.5), 450**) – Начало года 07:30.

где,'\*' – актуальные день/месяц/год

AddHours(-7.5), 450 – Сдвиг относительно основной функции

(-7.5), 450 – Применяется для задачи границы коммерческих суток (-7.5 – часы; 450 – минуты от значения границы)

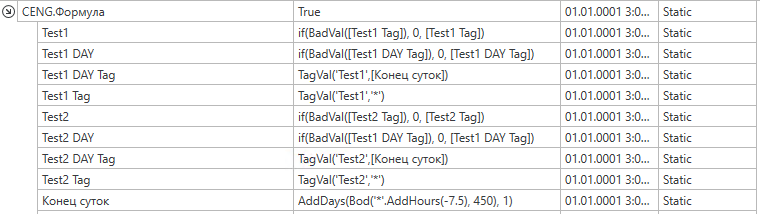
В случае минутного/часовго/суточного/месячного/годового сдвига метки относительно границы:

**AddDays**(Bod('\*'.AddHours(-7.5), 450)**, 1**) – Сдвиг метки на 1 сутки вперёд;

**AddDays**(Bod('\*'.AddHours(-7.5), 450)**,** **-1**) – Сдвиг метки на 1 сутки назад.

Аналогично выполняется для AddMinutes, AddHours, AddMonths, AddYears.

После настройки метки в атрибуте в дальнейшем на неё можно ссылаться через квадратные скобки [Метка времени] например, [Конец суток]



## 1.5.3 Проверка на bad. Функция If в связке с BadVal

Воспользуемся функциями **if** и **BadVal** для определения качества значений и действий функции в случае прохождения проверки.

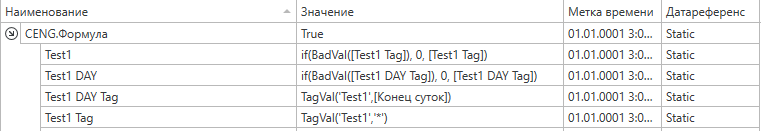
Создаём вложенный в CENG.Формула атрибут со следующим синтаксисом

**if**(**BadVal**([Атрибут с функцией TagVal]), **Действие-1**, **Действие-2**))

**If** – проверка условия

**BadVal** – функция проверки плохого качества значений

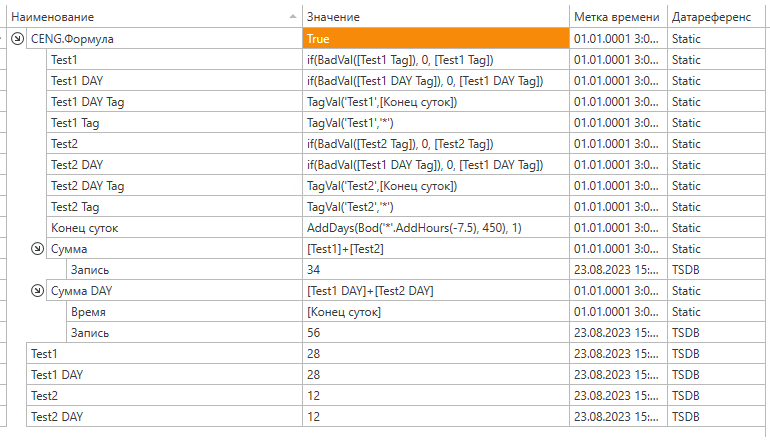
**Действие-1** – в случае прохождения проверки, как правило **0** (в некоторых случаях **NoOutput**)

**Действие-2** – в случае провала проверки, т.е. качества good в теге. По умолчанию [Имя атрибута Tagval]  


## 1.5.4 Простые суммы и разности

В качестве примера рассмотрим расчёт суммы (разность выполняется аналогично)

Составив иерархию из атрибутов TagVal и их проверки на Bad элемент примет следующий вид:



Атрибут **Сумма** – Состоит из ссылок на заранее созданные вложенные атрибуты проверки на Bad ([Test1],[Test2]) и математических действий с ними;

Аналогично **Сумма DAY**,но уже с атрибутами взятыми по метке времени

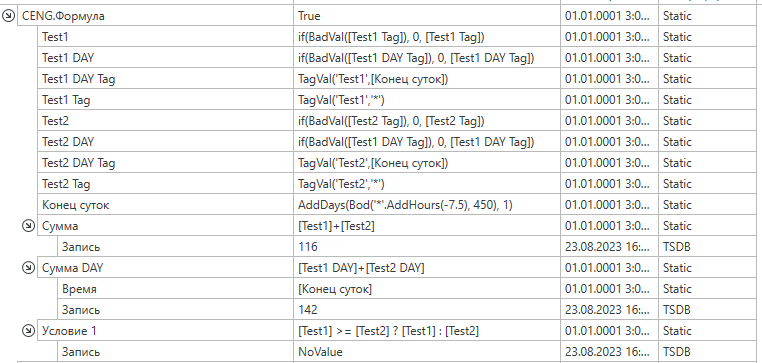
Атрибут **Время** – Обозначает метку времени на которую произойдёт запись результата

Атрибут **Запись** – Тег в который записывается результат расчёта

## 1.5.4 Расчёт с условиями

С помощью условий возможно задать несколько вариантов выполнения расчёта.

Рассмотрим на примере:



Атрибут **Условие 1** состоит из:

**[Test1] >= [Test2]** – Задание условия;

**?** – проверка на соответствие;

**[Test1] : [Test2]** – Варианты при выполнении/невыполнении условия.

Также возможно введение дополнительных действий в одном атрибуте через последовательную проверку:

([Test1]+[Test2]+[Test3]) == 0 ? NoOutput : (([Test1] >= [Test2] && [Test1] >= [Test3]? [Test1] : ([Test2] >= [Test1] && [Test2] >= [Test3]? [Test2] : ([Test3] >= [Test2] && [Test3] >= [Test1]? [Test3] : [Test1]))))

[Test1]+[Test2]+[Test3]) == 0 ? NoOutput – Проверка на 0, в случае прохождения результат NoOutput, иначе следующее действие

[Test1] >= [Test2] && [Test1] >= [Test3]? [Test1] – Проверка на наибольшее значение, в случае прохождения результат Test 1, иначе следующее действие

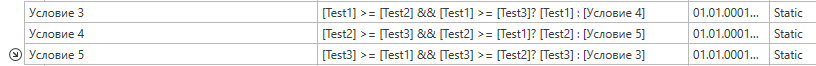
[Test2] >= [Test1] && [Test2] >= [Test3]? [Test2]– Проверка на наибольшее значение, в случае прохождения результат Test 2, иначе следующее действие

[Test3] >= [Test2] && [Test3] >= [Test1]? [Test3] : [Test1] – Проверка на наибольшее значение, в случае прохождения результат Test 3, иначе Test1

Расчёт будет выполнять проверки до получения результата.

## 1.5.4 Цикличные расчёты

Применяется для взаимосвязи нескольких условий и получения одного результата. Структура выстраивается путём взаимных ссылок [….] расчётов друг на друга.Цикл будет выполняться до получения результата одного из условий, который будет общим для всех выражений



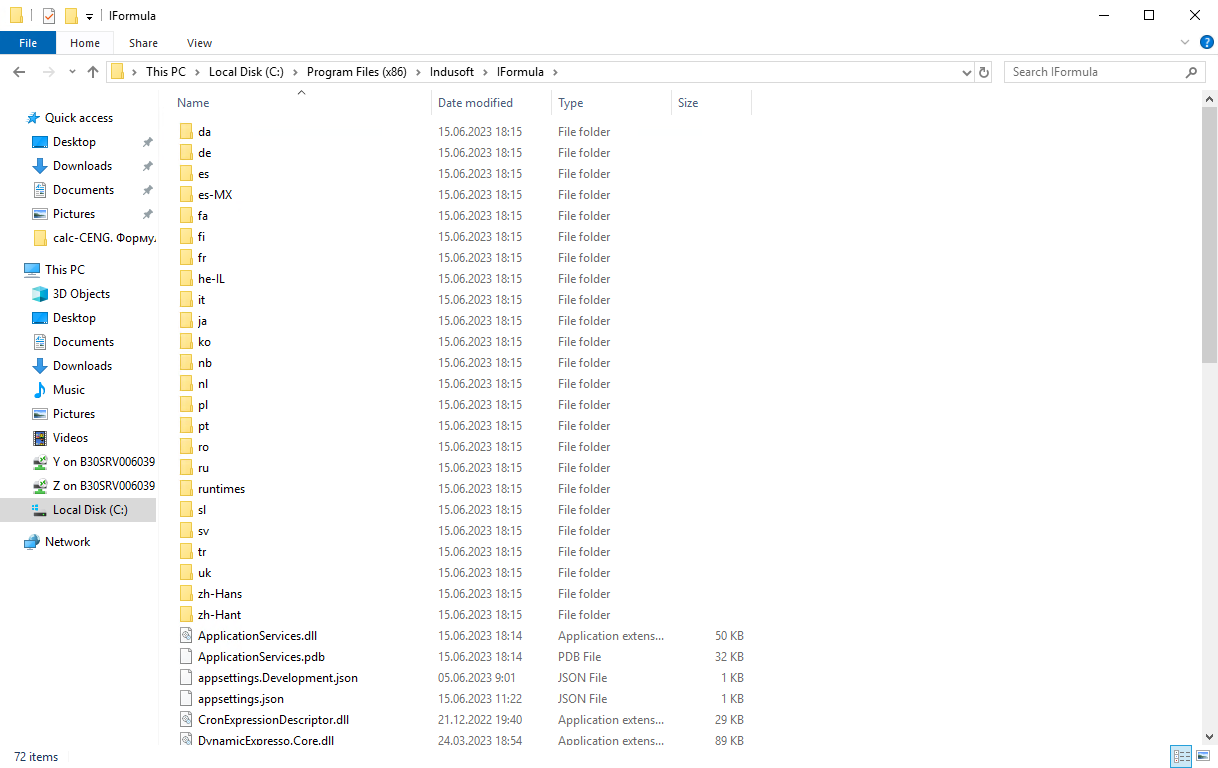
Результат можно вывести как в отдельный атрибут, так и в само выражение.





# Запуск приложения на сервере

Архив с приложением распаковываем на сервере:





## Настройка файлов конфигурации

### appsettings.json

DefaultConnection- строка подключения к БД I-DS\P,

TSDBAddress- адрес Tsdb WebApi,

TSDBLogin- логин Tsdb WebApi,

TSDBPassword- пароль Tsdb WebApi.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

### NLog.config

В секции rules можем выбрать/ добавить правила записи определённых логов конкретного уровня:

ApplicationServices.Calculator- логи расчётов,

ApplicationServices.Services и IFormula- логи приложения,

ApplicationServices.Scheduller – логи планировщика (по расписанию и триггеру).

В секции targets указаны места, куда записываются логи из соответствующих rules, а так же правила архивирования:

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

### Nodes.json

Данным конфигурационным файлом настраивается перечень узлов расчётов (можно добавлять необходимое количество через запятую):

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

SearchAttribute – атрибут, по которому в узел добавляется элемент, в который вложены расчётные атрибуты и значение которого true:

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание,

SearchModel – модель, в которой ищутся элементы ( \* - все модели),

SearchTemplate –шаблон элементов, которому должны соответствовать элементы ( \* - все шаблоны),

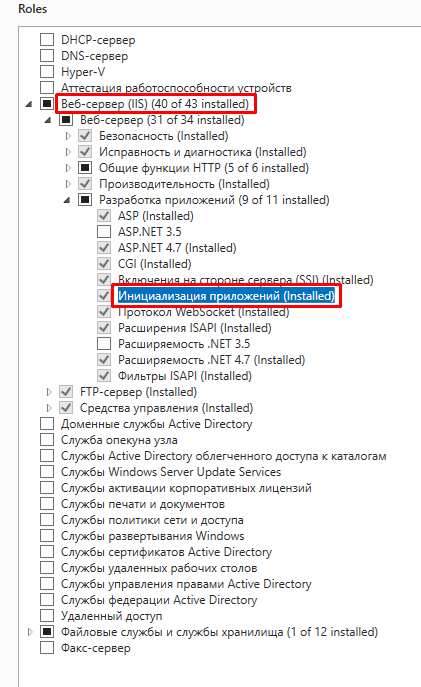
cronExpression – выражение планировщика по расписанию:  
<minute> <hour> <day-of-month> <month> <day-of-week> <command>

Более подробная информация о cron по ссылке:  
<https://losst.pro/nastrojka-cron>

https://www.baeldung.com/cron-expressions  
  
Пример: "0 0/1 \* ? \* \*" означает каждая минута в 0 секунд.

## 2.2 Настройка окружения

На сервере должна быть добавлена роль Веб-сервер IIS, а так же отдельно Инициализация приложений:



Должен быть установлен компонент .NET Core dotnet-hosting версии 6.0.15.

В IIS Manager во вкладке Application Pools необходимо добавить новый пул с соответствующими настройками:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Далее нажимаем Advanced Settings пула:

StartMode должен быть Always Running, Idle Time-out= 0 и Identity должна быть УЗ с правами к БД I-DS\P:

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Во вкладке Sites, выбрать Add Website, ввести название приложения, выбрать пул, указать каталог распакованного приложения и порт (8084 по умолчанию):

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

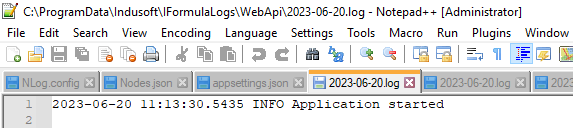
Автоматически созданное описание

В дополнительных настройках сайтах выставить значение True у параметра preloadEnabled:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Запустить пул приложения. При успешном запуске должна отобразиться следующая запись в логе WebApi:



# Управление

Управлять можно из браузера через интерфейс Swagger’а, а так же любым Http клиентом:  
Например локально перейдя по ссылке <http://localhost:8084/swagger/index.html>

## 3.1 Добавление узла расчётов (перезапуск при наличии)

Выбрать контроллер AddOrRestart, нажать Try it out, в поле Name необходимо записать имя атрибута узла расчётов из файла конфигурации Nodes.json приложения:

Изображение выглядит как текст, число, линия, программное обеспечение

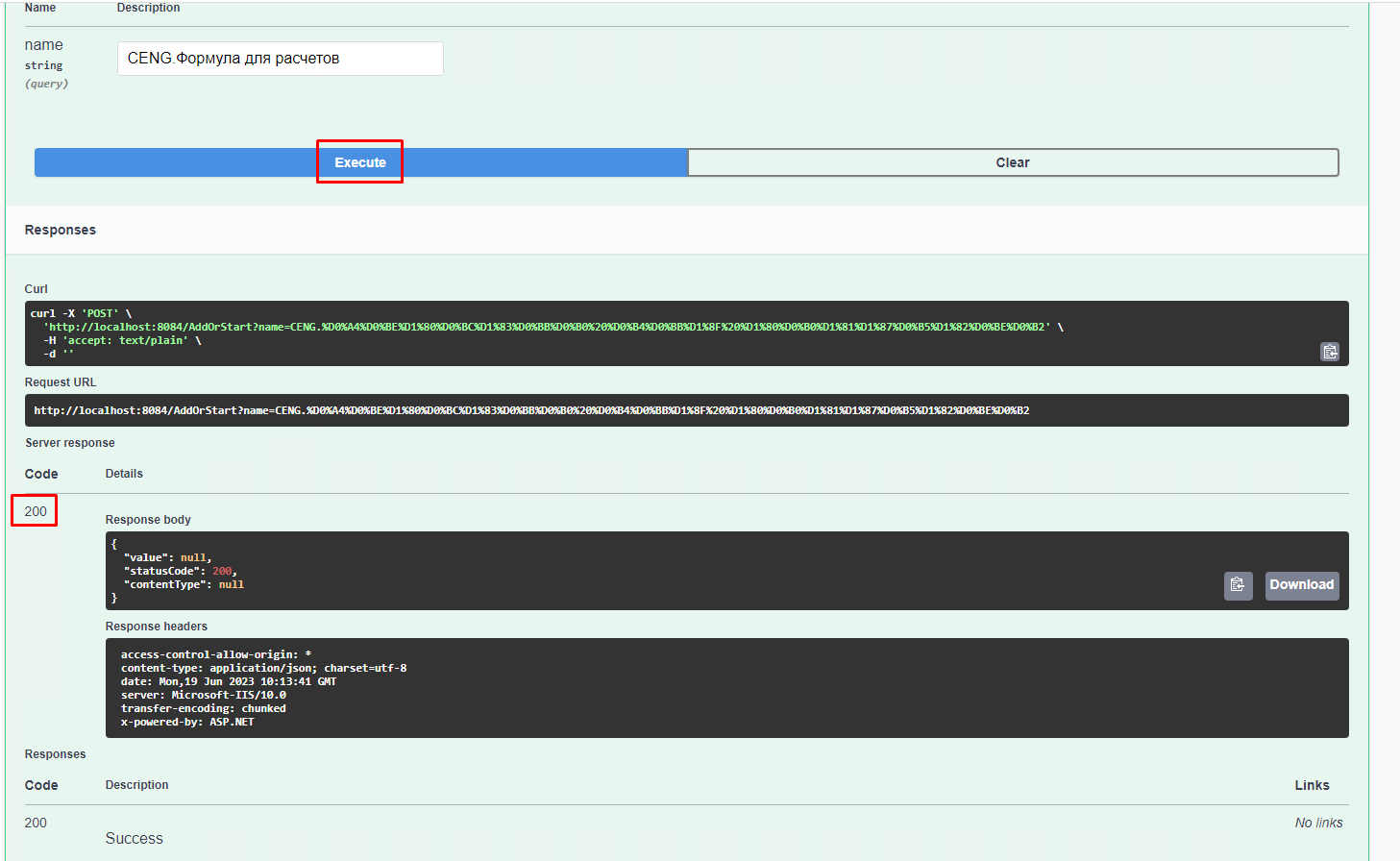
Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

При наличии указанного узла расчётов в программе он будет перезапущен (конфигурация будет считана заново).

Успешное добавление/перезапуск узла выдаст код ответа 200:



А также появится соответствующая запись к логе WebApi:



В случае ошибки код ответа будет 400 с соответствующим сообщением об ошибке:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание



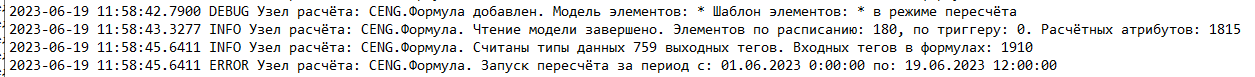
## 3.2 Пересчёт узла расчётов

Выбрать контроллер RecalcNode, нажать Try it out, в поле Name необходимо записать имя атрибута узла расчётов из файла конфигурации Nodes.json приложения, в поля startTimeLocal и entTimeLocal необходимо вписать строкой метки времени начала и окончания пересчёта соответственно в местном часовом поясе. Успешный запуск пересчёта выдаст код ответа 200:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

При старте и окончании пересчёта в логе WebApi будет сделана соответствующая запись:



## 3.3 Удаление узла расчётов

Выбрать контроллер StopAndDeleteNode, нажать Try it out, в поле Name необходимо записать имя атрибута узла расчётов из файла конфигурации Nodes.json приложения. Успешная остановка узла выдаст код ответа 200 с соответствующей записью в лог:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**Внимание!** Остановка узла расчёта выполнит остановку соответствующих задач текущих расчётов в приложении, а так же удалит соответствующую часть конфигурации в файле Nodes.json.

# Планируемые доработки функционала

## 4.1 Вывод промежуточных результатов онлайн

Реализовать возможность вывода результатов расчёта промежуточных атрибутов (тех, которые никуда не записываются и хранятся в кеше приложения) в режиме онлайн. Данные атрибуты обычно используются для предварительной подготовки части данных для главного вычисления, записываемого в тег:

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

В настоящий момент результат расчёта выводится только в лог файл. Планируется сделать http контролер получения результата расчёта таких атрибутов в целях возможности реализации инструмента отладки IFormula.

## 4.2 Реализовать прототип конфигуратора IFormula

Предварительно на Windows Forms реализовать клиент IFormula с возможностью просмотра результатов расчётов напротив атрибутов используя WebApi IFormula (по аналогии с AF), т.е. в окно с атрибутами элемента добавить столбец «Результат расчёта».

Так же планируется реализовать все функции управления из п.3 в данном клиенте.

## 4.3 Реализовать при необходимости новые вычислительные функции

Список в документации будет актуализироваться по мере доработок IFormula