Рабочая тетрадь к курсу Systeme Platform «SEP-11-el. Базовый курс разработки»

(Разработка проекта и обслуживание систем автоматизации на базе Systeme Platform)

Версия документа: 2023-02-15

Оглавление

Введение

Данная рабочая тетрадь создана с целью упрощения изучения компонентов Systeme Platform для инженеров и разработчиков с любым уровнем опыта в АСУ ТП (от начинающего, до продвинутого). Результатом прохождения данного материала является простой проект, который охватывает базовые возможности основных компонентов Systeme Platform.

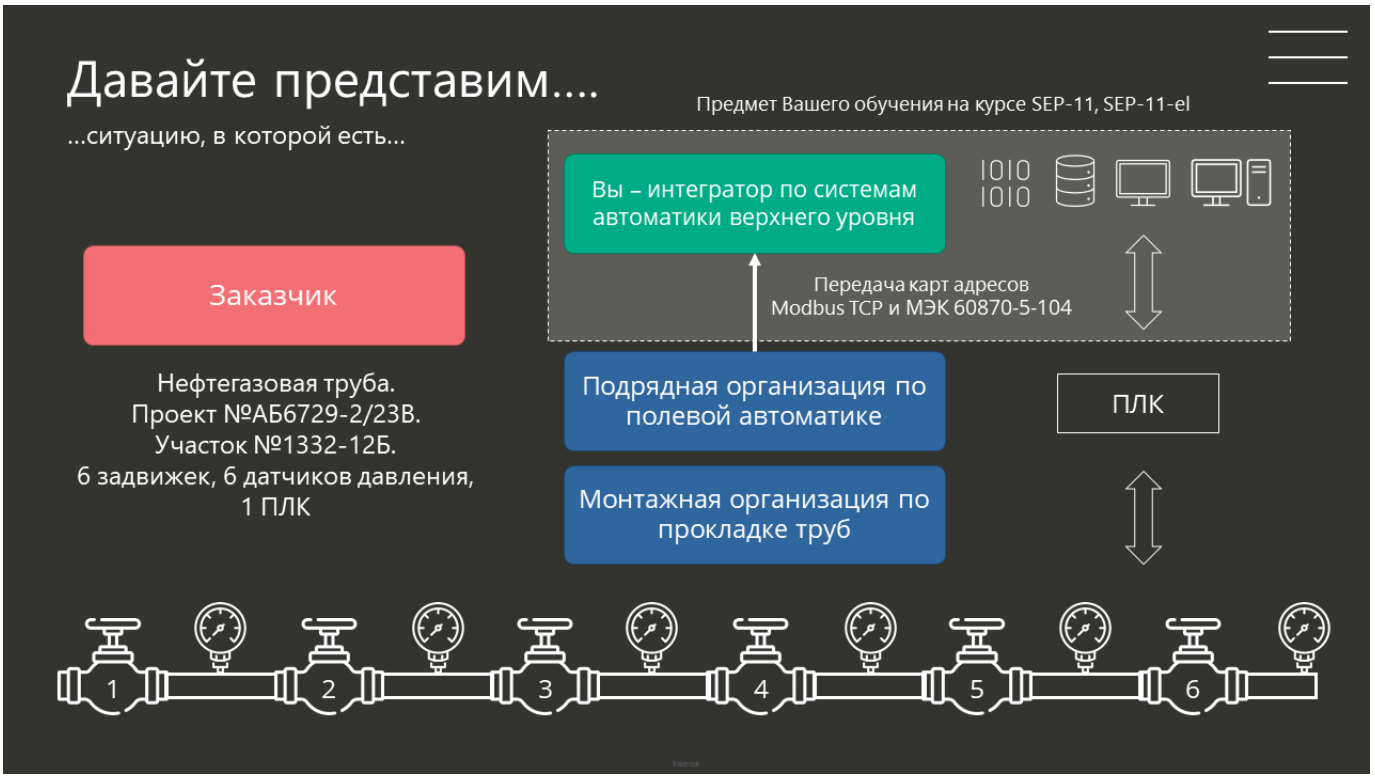
Содержание тетради полностью дублирует видеокурс обучения на образовательной платформе «Learning 4U» Центра обучения Systeme Electric, а его задания повторяют содержание видеообучения, чтобы учащийся мог выбрать наиболее подходящий для него формат, а также иметь возможность повторить курс обучения в случае отсутствия доступа к видеоматериалам.

В рамках программы обучения вы разработаете и изучите условный демонстрационный проект «Нефтегазовая Труба», включающий в себя разработку резервируемой архитектуры, работу с базами данных, настройку человеко-машинного интерфейса, графиков и аварий, а также настройку прав пользователей и многое другое.

Построение демонстрационного проекта будет происходить от простого к сложному. В процессе разработки проекта необходимо будет периодически переключаться между компонентами SCADA и обращаться к средствам отладки.

Для прохождения курса обучения вам будут предоставлены две виртуальные машины: с ОС Windows и с ОС Astra Linux Смоленск. Также всем учащимся будет предоставлена дополнительная виртуальная машина, на которой развернут проект сервера для имитации логики работы контроллера и других устройств полевого уровня управления.

К учебному проекту также приложены дополнительные инструкции, которые можно найти в документе *«Краткий конспект слушателя многодневного обучения»*

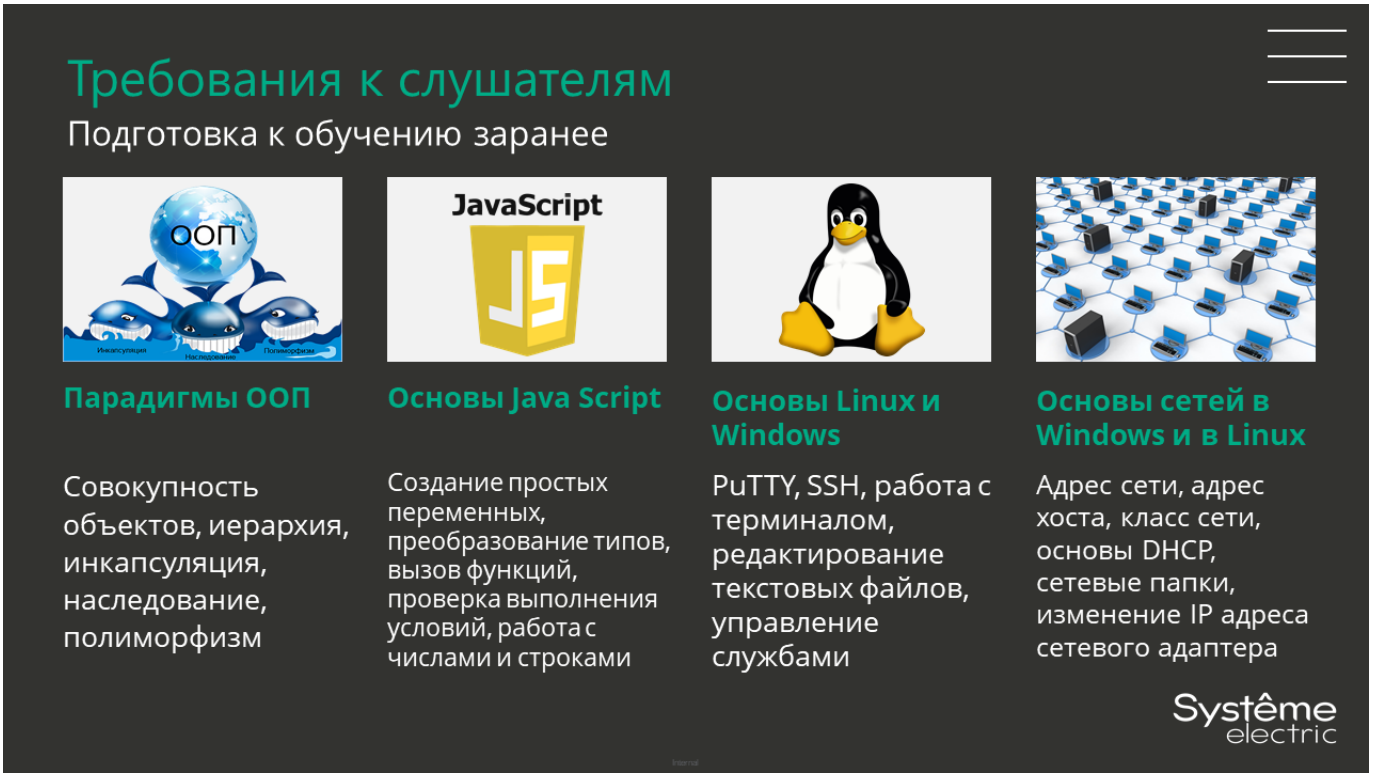


Целевая аудитория

- SCADA-инженеры и специалисты по развертыванию систем диспетчеризации и управления
- Инженеры АСУ
- Операторы и инженеры обслуживающих компаний
- Системные архитекторы и специалисты по интеграции программного обеспечения
- Коммерсанты и специалисты по продвижению программного обеспечения

Требования к слушателям

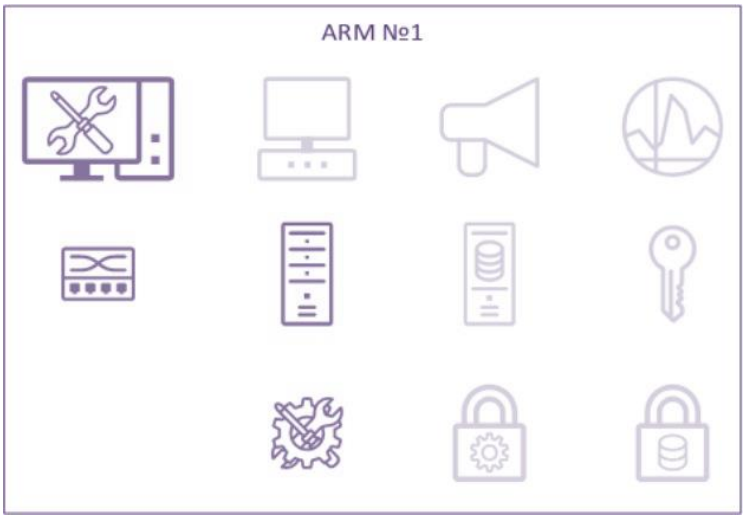
- Успешное завершение вводного вебинара SEP-01 «Systeme Platform - единое ПО для любых задач автоматизации»
- Понимание основных принципов развертывания SCADA систем
- Базовые знания по темам: Парадигмы ООП; Основы Java Script; Основы Linux и Windows; Основы сетей в Windows и в Linux
  Примечание: *Базовые знания по указанным выше темам, а также инструкция по взаимодействию с виртуальными машинами описаны в видео 00-1 – 00-5 в рамках части 0 предоставленной программы обучения на учебной платформе «Learning 4U!»*



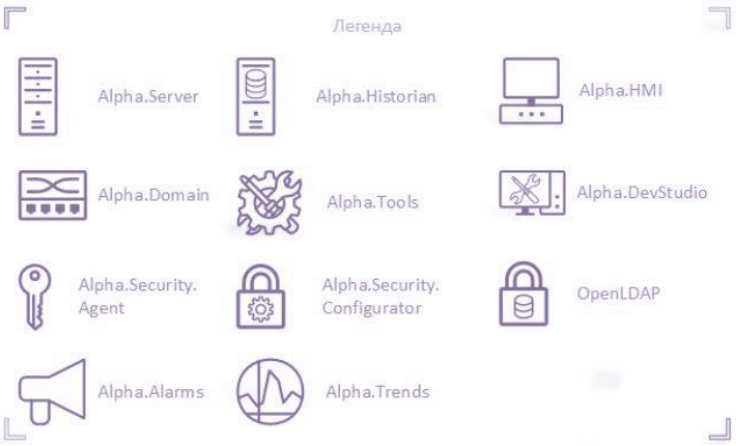
Урок 01. Установка и настройка требуемых компонентов на Windows

Установка компонентов Systeme Platform

Для начала необходимо установить компоненты (выделенные фиолетовым) согласно схеме:



Данная схема будет дополняться компонентами и связями по мере прохождения материала. Процесс установки и настройки компонентов описан в документации и видеокурсе обучения. Компоненты не требуют сложной настройки на начальном этапе. Добавляемые элементы будут отмечены зелёным.



Требования к рабочему месту

> ***Примечание**: данные требования необходимы только в случае прохождения обучения на собственной машине, а также для понимания первоначальных действий по настройке машин на объекте.*
>
> *В рамках прохождения обучения в Systeme Electric вам будут предоставлены заранее преднастроенные виртуальные машины.*

Рабочие места должны быть оборудованы:

- Двумя машинами с ОС Windows 10 (желательно с актуальным пакетом обновлений);
- Виртуальной машиной с ОС Astra Linux Смоленск;
- Не менее 4 ГБ оперативной памяти;
- Все компьютеры должны быть в одной сети;
- Отключен брандмауэр;

На компьютерах предварительно необходимо установить сторонние компоненты:

- .NET Framework 3.5
- .NET Framework 4.6.1 и выше
- Microsoft Visual C++ 2013 Redistributable (x64)
- Microsoft Visual C++ 2013 Redistributable (x86)
- Microsoft Visual C++ 2015-2019 Redistributable (x64)
- Microsoft Visual C++ 2015-2019 Redistributable (x86)
- OPC .NET API 2.00 Redistributables актуальной версии
- OPC Core components Redistributables (x64) актуальной версии
- OPC Core components

Дополнительно:

- Браузер Google Chrome
- Notepad++
- PuTTY для подключения по протоколу SSH
- WinSCP опционально

Совет по установке сторонних компонентов:

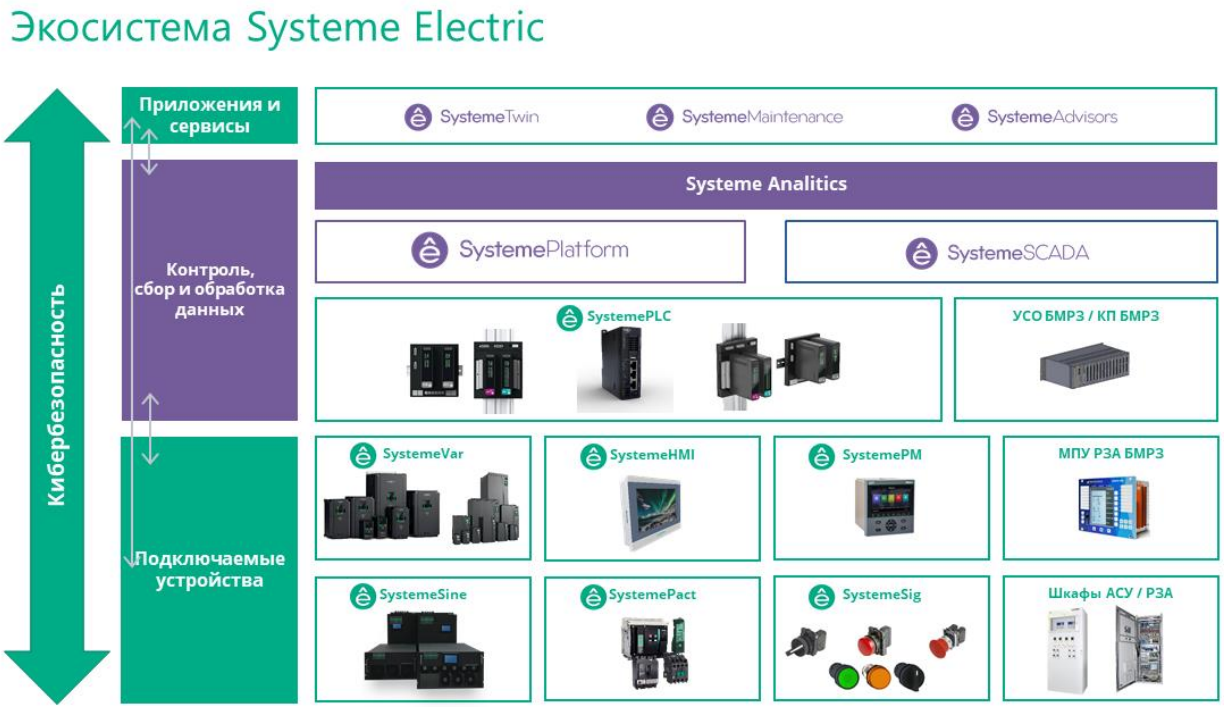
Можно сначала включить .NET Framework 3.5. После этого можно запустить дистрибутив *SePlatform.Alarms* или *SePlatform.Trends*. В состав данных дистрибутивов входят основные сторонние компоненты, которые необходимы платформе Systeme Platform. После установки *SePlatform.Alarms* или *SePlatform.Trends* можно удалить, сторонние компоненты останутся.

Настройки системы

Для корректной работы *SePlatform.Historian* необходимо настроить блокировку страниц памяти. Зайти в Панель управления → Просмотр: Мелкие значки → Администрирование → Локальная политика безопасности → Локальные политики → Параметры безопасности → Назначение прав пользователя → Блокировка страниц в памяти → Добавить пользователя или группу → Дополнительно → Поиск → Добавить пользователей Все и АНОНИМНЫЙ ВХОД → Нажать ОК.

Об АСУ ТП

АСУ ТП (Автоматизация систем управления технологическим процессом) – комплекс управляющих компьютерных устройств и их объединений с целью управления разнообразными процессами. Используется преимущественно на промышленных предприятиях (в т.ч. энергетической, нефтегазовой сферы, производства), а также для управления транспортом, инженерными системами и т.п.



Нижний уровень (полевой уровень, уровень подключаемых устройств) – это уровень того физического оборудования, которое находится прямо на объекте. Этим оборудованием необходимо управлять и мониторить его работу. Это уровень датчиков, измерительных устройств, контролирующих управляемые параметры, а также исполнительных устройств, воздействующих на эти параметры (краны, задвижки, системы вентиляции, насосы, двигатели, и т.п.). На этом уровне осуществляется согласование сигналов датчиков с входами устройства управления, а вырабатываемых команд с исполнительными устройствами.

Средний уровень (контроль, сбор и обработка данных, уровень управления оборудованием) – это уровень контроллеров.

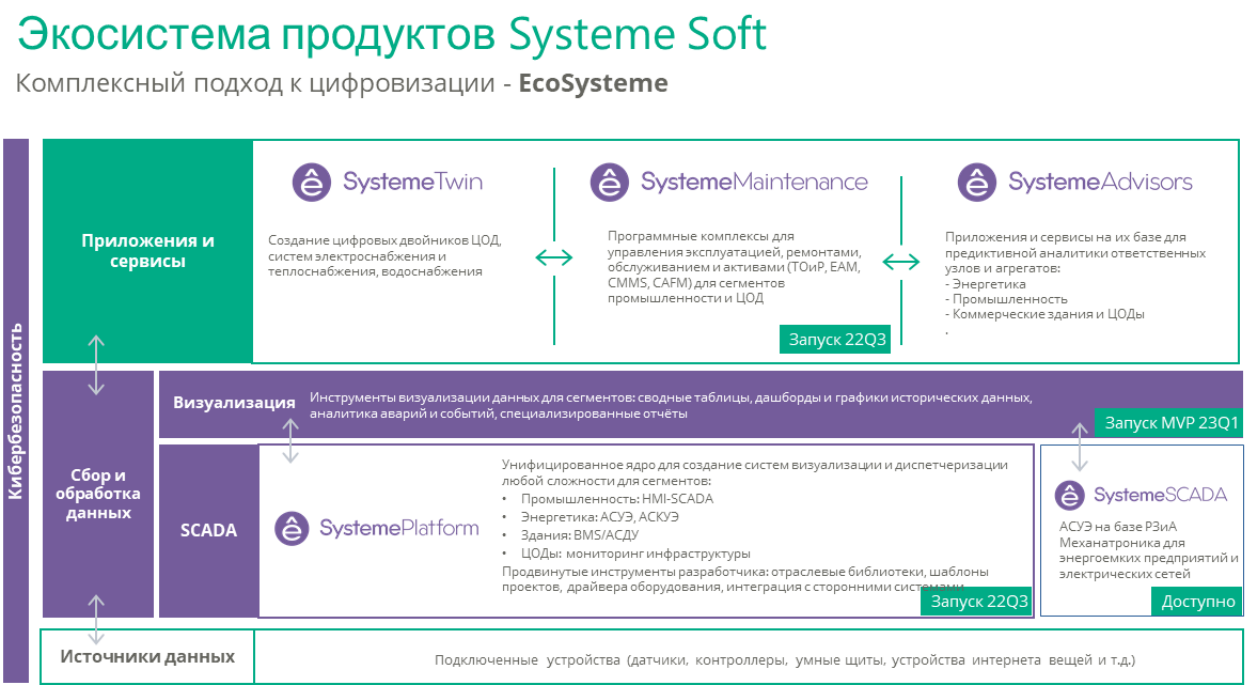
ПЛК (Программируемый логический контроллер, PLC) получает информацию с контрольно-измерительного оборудования и датчиков о состоянии технологического процесса и выдает команды управления в соответствии с запрограммированным алгоритмом управления на исполнительные механизмы. По сути, работает с нижним уровнем, обменивается информацией с верхним уровнем.

Верхний уровень (уровень операторских и диспетчерских станций) – это комплекс программно-аппаратных средств, выполняющих функции сбора и предварительной обработки данных от датчиков технологического процесса о состоянии оборудования и исполнительных механизмов. Здесь происходит контроль данных на достоверность, обеспечение человеко-машинного интерфейса (HMI). С верхнего уровня оператор получает информации о текущем состоянии технологического процесса в виде мнемосхем, графиков, таблиц, сигнализаций. Это уровень SCADA и MES систем, который работает со средним уровнем либо со сторонними системами верхнего уровня для сбора, передачи, хранения, обработки и анализа данных.

О Systeme Platform

Архитектура

Программный комплекс Systeme Platform состоит из компонентов, используемых для разработки, исполнения и сопровождения проектов автоматизации. Функциональность платформы охватывает верхний уровень АСУ.



Ключевая единица платформы – домен (совокупность вычислительных средств для исполнения проекта автоматизации). В него входят серверные компоненты: *SePlatform.Server* (сервер ввода-вывода), *SePlatform.Historian* (сервер истории).

К подсистемам инфраструктуры Systeme Platform платформы относится *SePlatform.Security* (безопасность и разграничение прав пользователей), *SePlatform.Diagnostics* (комплексные функции диагностики,
мониторинга и аудита) и *SePlatform.Licensing* (обеспечивает лицензирование продуктов).

SePlatform.AccessPoint – единая точка доступа, объединяющая серверные (в т.ч. резервируемые) компоненты и удалённые домены платформы. Служит точкой предоставления данных сторонним системам верхнего уровня, и пользовательской среде Systeme Platform платформы.

Пользовательская среда для работы с визуальной частью проекта, обеспечивающая человеко-машинный интерфейс: *SePlatform.HMI* (мнемосхемы), *SePlatform.Alarms* (события и тревоги) и *SePlatform.Trends* (графики).

Среда разработки и управления для создания, тестирования и отладки приложений включает *SePlatform.DevStudio* и *SePlatform.Tools*. Большая часть компонентов Systeme Platform платформы кроссплатформенная.

1. О SePlatform.Server

Основной задачей *SePlatform.Server*, как компонента Systeme Platform, является выполнение функций сервера ввода/вывода. *SePlatform.Server* выполняет сбор технологических данных с коммуникационных устройств в ходе мониторинга контролируемых объектов. На основе полученной информации осуществляется контроль над технологическим процессом. Управление может происходить как по команде оператора при передаче собранных данных на верхние уровни АСУ ТП, так и по встроенным алгоритмам. *SePlatform.Server* является шлюзом для работы SCADA-системы с устройствами ввода/вывода
информации. Одновременно сервер может поддерживать соединение с несколькими промышленными контроллерами. Возможна установка нескольких экземпляров *SePlatform.Server* на одном компьютере. *SePlatform.Server* работает в системе в виде службы. Для запуска/остановки сервера достаточно запустить/остановить службу ***SePlatform.Server***.

Архитектура

Основными составляющими *SePlatform.Server* являются программные модули. Набор модулей подбирается в соответствии с поставленной задачей.

Задачи, решаемые сервером:

- Сбор данных с устройств и сторонних серверов:
  - IEC 104 Master (спецификация ГОСТ Р МЭК 60870-5-104)
  - IEC 61850 Client (спецификация МЭК 61850)
  - Modbus TCP Master (спецификация Modbus TCP)
  - Modbus RTU Master (спецификация Modbus RTU)
  - OPC DA Client (спецификация OPC DA)
  - OPC HDA Client (спецификация OPC HDA)
  - OPC UA Client (спецификация OPC UA)
  - Hub Module (TCP, Файловый интерфейс)
  - SQL Connector (SQL)
  - SNMP Manager (SNMP (v1, v2, v3), ICMP)
  - Syslog Server (Syslog)
  - NetDiag (сети TCP/IP)
- Предоставление данных клиентам:
  - IEC 104 Slave (спецификация ГОСТ Р МЭК 60870-5-104)
  - Modbus TCP Slave (спецификация Modbus TCP)
  - Modbus RTU Slave (спецификация Modbus RTU)
  - OPC DA Server (спецификация OPC DA)
  - OPC HDA Server (спецификация OPC HDA)
  - OPC AE Server (спецификация OPC AE)
  - OPC UA Server (спецификация OPC UA)
  - TCP Server Module (TCP, Файловый интерфейс)

Ядро является центральным компонентом сервера для реализации инфраструктуры сервера, интерфейсов работы с модулями, сигналами и их свойствами, остальными подсистемами. Основные функции ядра:

- Управление работой модулей;
- Управление резервированием;
- Запись и чтение данных из БД;
- Отправка и принятие уведомление об изменении сигналов;
- Пересчёт из физических значений в инженерные и обратно (линейная зависимость, и с изломом);
- Выполнение по событиям, таймеру, расписанию;

Резервирование:

Для повышения надёжности и минимизации времени отсутствия основных функций системы сбора при невозможности ПОЛНОГО ДУБЛИРОВАНИЯ серверов используется ГОРЯЧЕЕ РЕЗЕРВИРОВАНИЕ. Горячим резервированием управляет Модуль резервирования.

Логическая обработка данных:

Промежуточной обработкой данных занимается Logics Module. Вычисления, проводимые модулем, вынесены на уровень ядра.

Возможности модуля логики:

- Пересчёт значений из физических в инженерные и обратно;
- Пересчёт по формуле (логические, арифметические, битовые операции);
- Выполнение алгоритмов по событию, таймеру, расписанию;
- Вызов функций из внешних динамических библиотек (*.dll);
- Перехват генерируемых событий и тревог;
- Опциональное изменение свойств сигнала VQT (Модуль Write VQT);
- Разбор буфера для выделения кода технологического объекта и кода события

(Модуль Data Buffer).

Генерация событий и тревог:

На основе полученных и обработанных данных, сервер может по заранее определённым правилам генерировать и предоставлять пользователям сообщения о событиях и тревогах (Модуль OPC AE Server).

Возможности:

- Генерация событий по спецификации OPC AE;
- Предоставление по OPC DA, UA, AE;
- Сохранение событий в историю;
- Отправка информации о событиях по электронной почте;

Прочие возможности *SePlatform.Server*:

- Сохранение текущих значений сигналов в файлы срезы XML (Модуль Snapshot) и бинарные файлы (Модуль FS Generator);
- Диагностика сетевых устройств (Модуль NetDiag);
- Предоставление данных для записи в сервер истории (Модуль History Module);

Сервисное обслуживание сервера:

- Редактирование конфигурации с помощью *SePlatform.DevStudio*;
- Просмотр статистики с помощью приложения *Статистика*;
- Просмотр журналов работы модуля с помощью приложения *Просмотрщик лога кадров*;
- Управление работоспособностью сервера и резервной пары с помощью приложения *Управляющий*;
- Для отладки используется *SePlatform.OpcExplorer* и *SePlatform.EventLogViewer*.

Работа с SePlatform.Server

Каждый экземпляр *SePlatform.Server* работает в системе в качестве службы. Сервер, как отдельный компонент, можно конфигурировать с помощью сервисного приложения *Конфигуратор*. Также можно конфигурировать сервер в составе проекта автоматизации с помощью *SePlatform.DevStudio*. В том числе, с помощью подсистемы *SePlatform.Domain* можно разворачивать конфигурации на экземпляры *SePlatform.Server*.

Сервисные приложения

В комплект поставки *SePlatform.Server* входят сервисные приложения для управления, конфигурирования, просмотра статистической информации. Данные приложения предназначены для установки на автоматизированных рабочих местах администраторов. Все сообщения о работе сервера и его компонентов фиксируются в журнал приложений ОС Windows.

Конфигуратор

Сервисное приложение *Конфигуратор* входит в состав клиентской части дистрибутива *SePlatform.Server*.

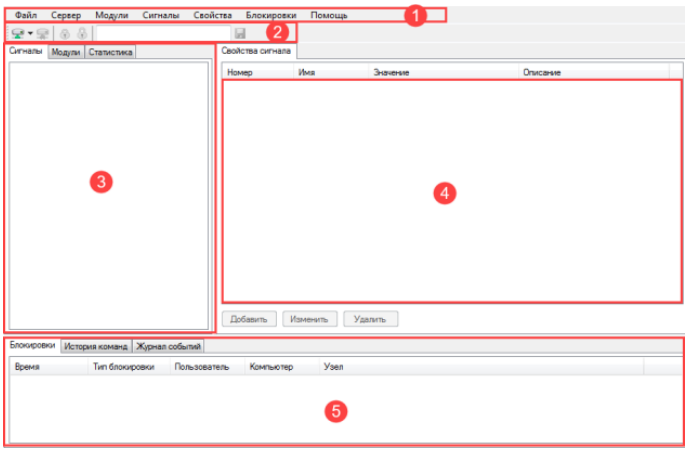
Приложение Конфигуратор можно запустить: Пуск → SePlatform → Конфигуратор. В новых проектах, созданных с помощью *SePlatform.DevStudio*, *Конфигуратор* не используется. *Конфигуратор* позволяет решать следующие задачи пользователя:

- Открытие и просмотр адресного пространства сервера;
- Конфигурирование дерева сигналов сервера;
- Формирование перечня свойств сигналов сервера;
- Конфигурирование и настройка модулей сервера;
- Сохранение созданной конфигурации в файл;
- Загрузка готовой конфигурации из файла;
- Отображение статистической информации сервера и журнала конфигурирования;

- Создание резервной копии текущей конфигурации;
- Создание и редактирование пароля для защиты конфигурации от несанкционированного доступа.

Состав окна *Конфигуратора*:

1. Главное меню;
2. Панель инструментов;
3. Область конфигурирования;
4. Область настраиваемых параметров;
5. Область логирования: блокировки, история команд и журнал событий.



Статистика

Сервисное приложение *Статистика* предназначено для просмотра статистических данных сервера ввода/вывода *SePlatform.Server*, сервера исторических данных *SePlatform.Historian* и сервера лицензирования *SePlatform.LicServer*. В статистике сервера можно увидеть количество лицензируемых тегов и модулей у готовой конфигурации.

Приложение *Статистика* можно запустить: Пуск → SePlatform → Статистика. Сервисное приложение *Статистика* позволяет:

- Подключаться к вышеперечисленным серверам по интерфейсу TCP/IP;
- Сохранять текущий набор статистических данных в файл;
- Загружать для просмотра файлы статистических данных.

Просмотрщик лога кадров

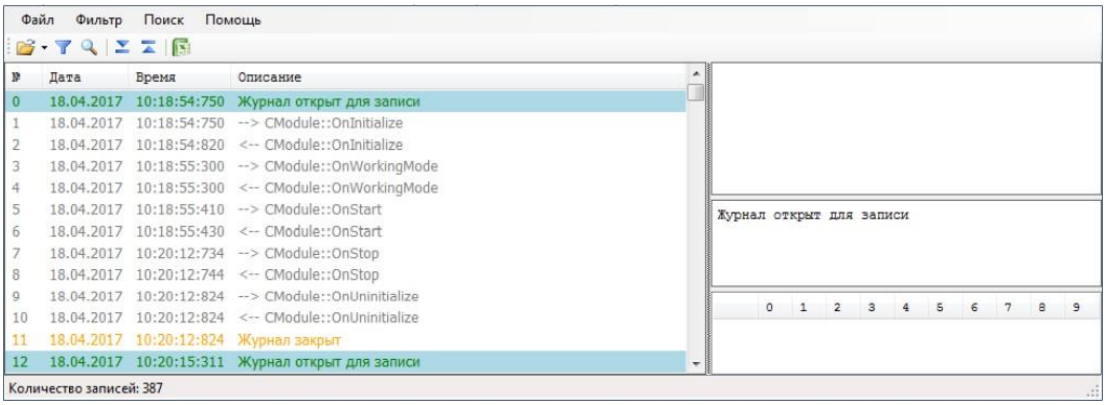
Сервисное приложение *Просмотрщик лога кадров* в составе дистрибутива *SePlatform.Server* предназначено для просмотра журналов работы модулей *SePlatform.Server*.

Приложение *Просмотрщик лога кадров* можно запустить: Пуск → SePlatform → SePlatform.Server → Просмотрщик лога кадров. Сервисное приложение *Просмотрщик лога кадров позволяет*:

- Отображать журнал работы модуля как в оперативном режиме, так и сохранённые журналы;
- Фильтровать и искать записи;
- Экспортировать сообщения о работе модуля в табличный файл.

Журнал позволяет отображать:

- Сообщения о работе модуля;
- Технологические данные, с которыми работает модуль.



2. Работа с SePlatform.DevStudio

SePlatform.DevStudio – это инструмент для построения проектов автоматизации (конфигураций *SePlatform.Server*). В основе построения проектов используется объектно-ориентированный подход.

SePlatform.DevStudio позволяет работать над несколькими проектами в команде благодаря возможности модульного построения проектов и развёртывать их с помощью подсистемы развёртывания *SePlatform.Domain*. *SePlatform.DevStudio* создаёт конфигурации для *SePlatform.Server* на основании
решения, в котором описывается информационная модель обмена данными между компонентами и логика их работы.

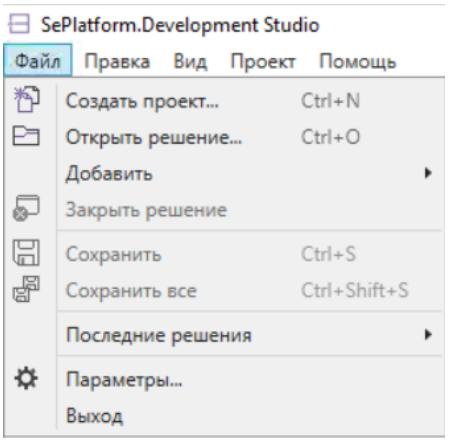
Знакомство с компонентами разработки. Развертывание простого проекта

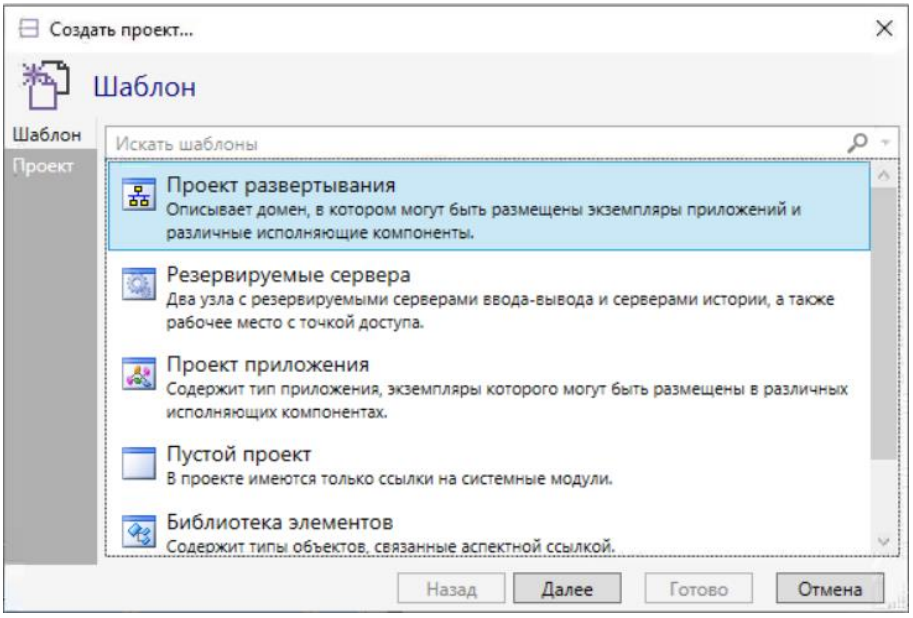
Для начала необходимо научиться настраивать простой проект развёртывания. В примере будем применять простую конфигурацию с одним сигналом на основной экземпляр *SePlatform.Server*.

Создание простой конфигурации:

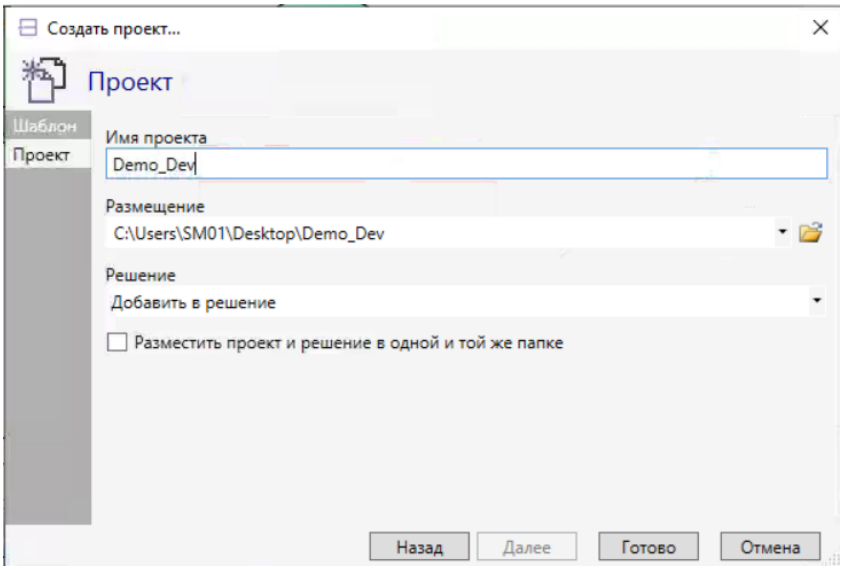
2.1 Запустите *SePlatform.DevStudio* Пуск → SePlatform → SePlatform.DevStudio.

2.2 Создайте проект развёртывания. Файл → Создать проект → Проект развертывания



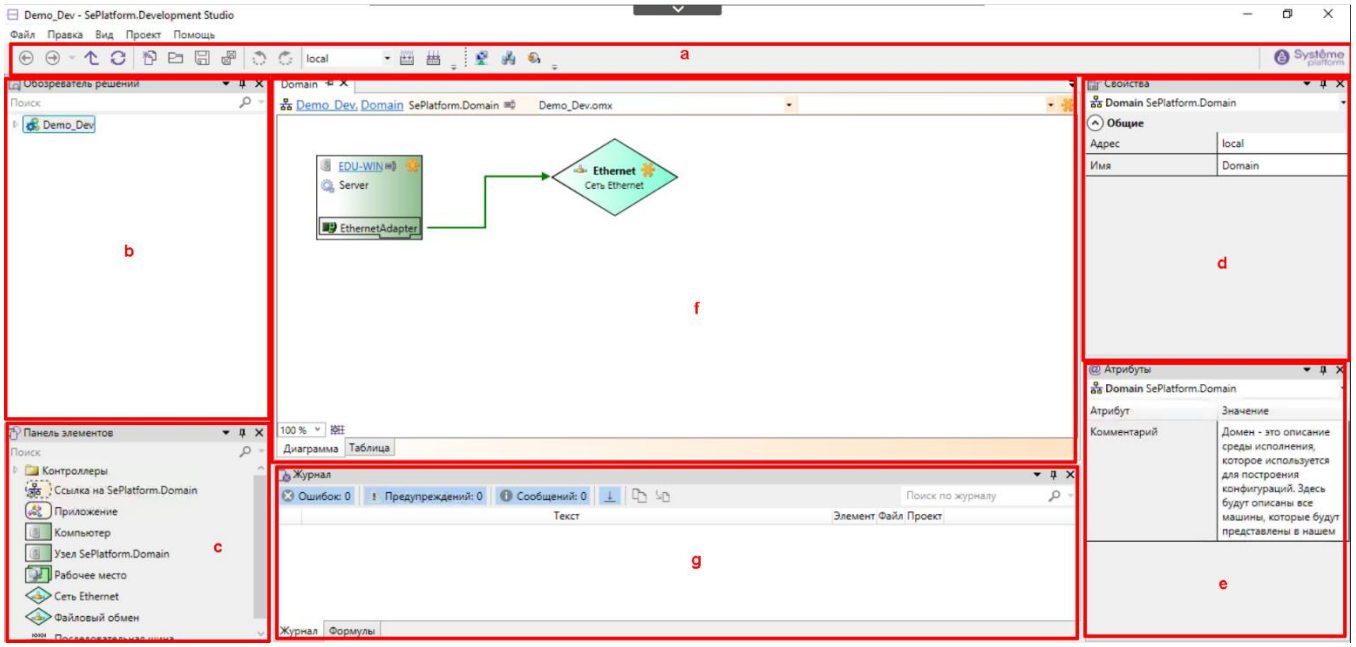


2.3 Задайте проекту имя Demo_Dev, затем нажмите «готово»



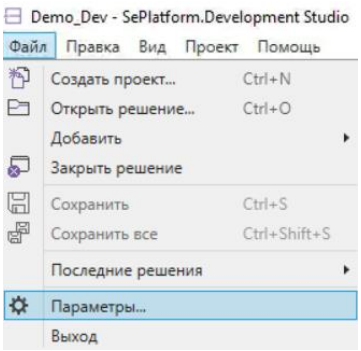
Перед Вами среда разработки, которая содержит:

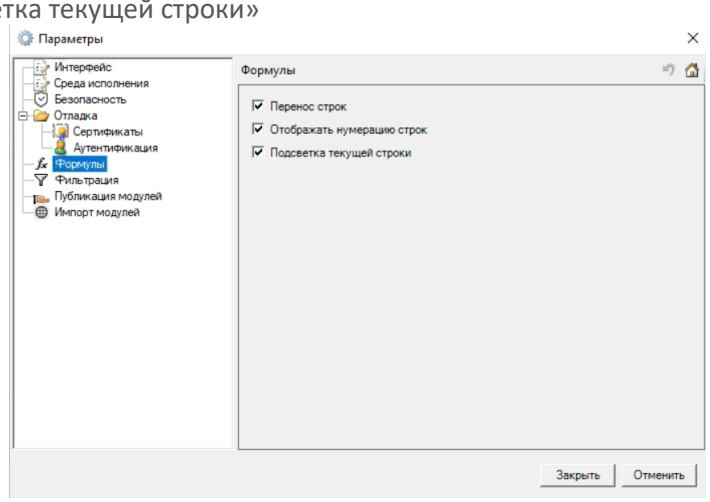
a. Панель управления;
b. Обозреватель решений (используется для навигации по проекту);
c. Панель элементов (содержит список элементов, которые можно расположить внутри того или иного места в проекте);
d. Панель со свойствами;
e. Панель атрибутов;
f. Область журнала и поля для ввода формул;
g. Рабочую область.



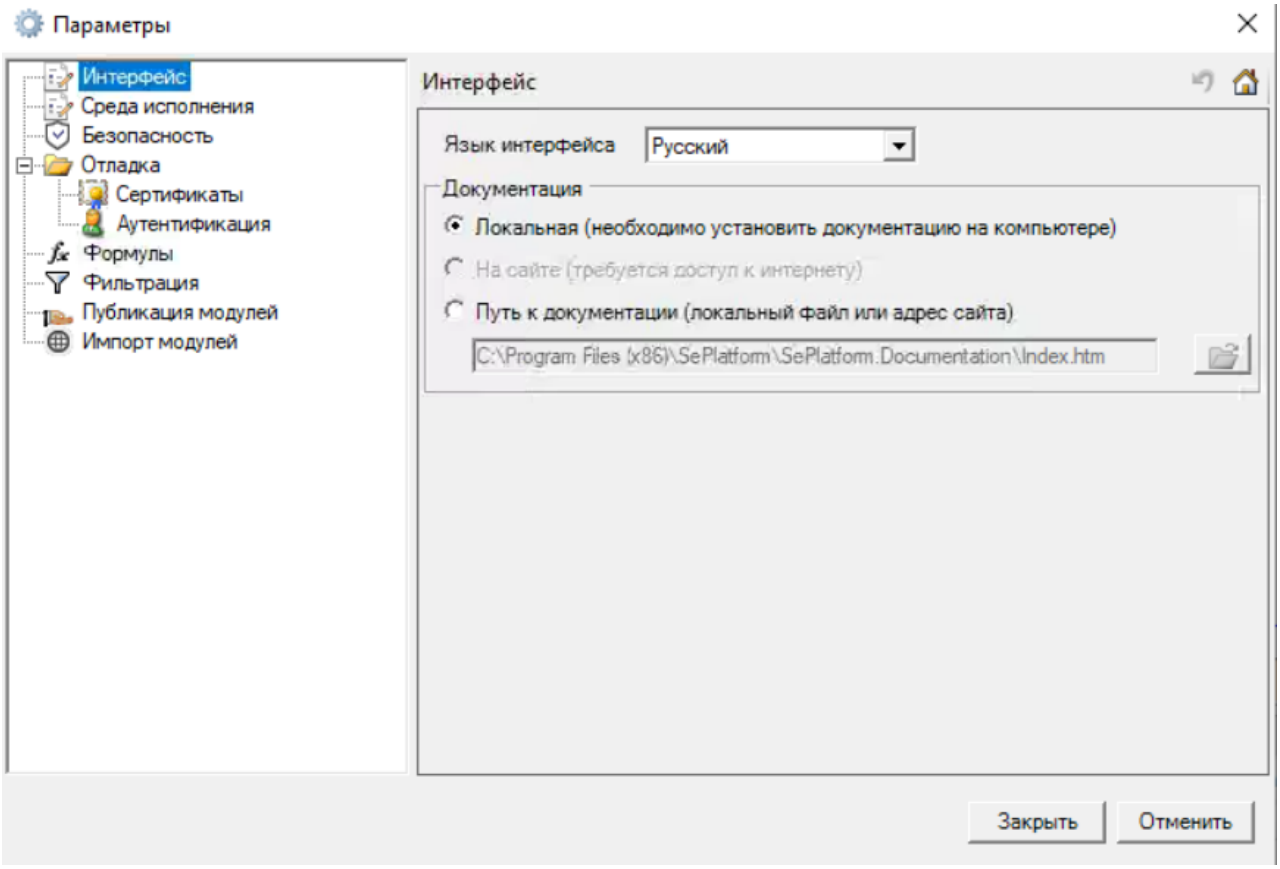
Подсказка: для удобства работы можно опционально произвести несколько настроек интерфейса:

Чтобы упростить работу с формулами нужно перейти Файл → Параметры, выбрать параметр «Формулы» и отметить два чек-бокса: «Перенос строк» и «Подсветка текущей строки»

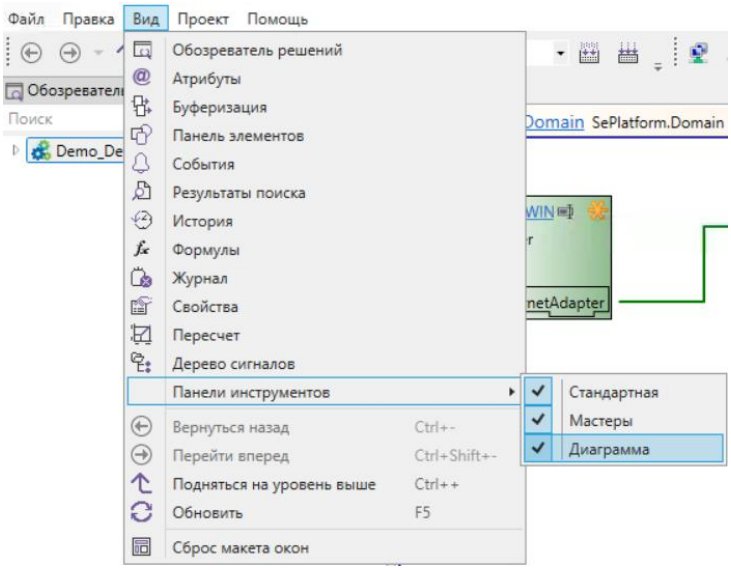




Чтобы сменить язык интерфейса нужно так же в меню Файл → Параметры выбрать «Интерфейс» и сменить язык на Русский, либо Английский (он сменится после перезапуска DevStudio)

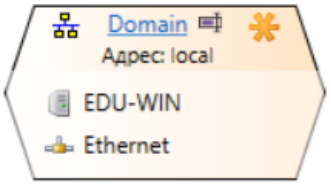


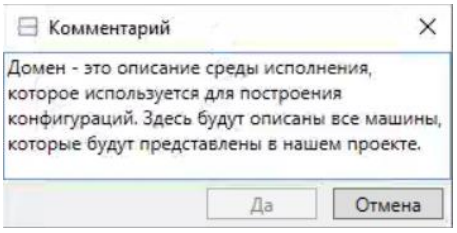
Для отображения инструментов выравнивания можно перейти в меню Файл → Панели инструментов и отметить галочкой «Диаграмма»



2.4 Для упрощения навигации и структуры в проекте можно использовать комментарии. Комментарии используются для удобства разработки проектов. Они отображаются только в проекте DevStudio. Желательно указывать комментарии для всех компонентов, используемых в проекте. Комментарий также можно посмотреть при наведении курсора на элемент, либо во вкладке Атрибуты.

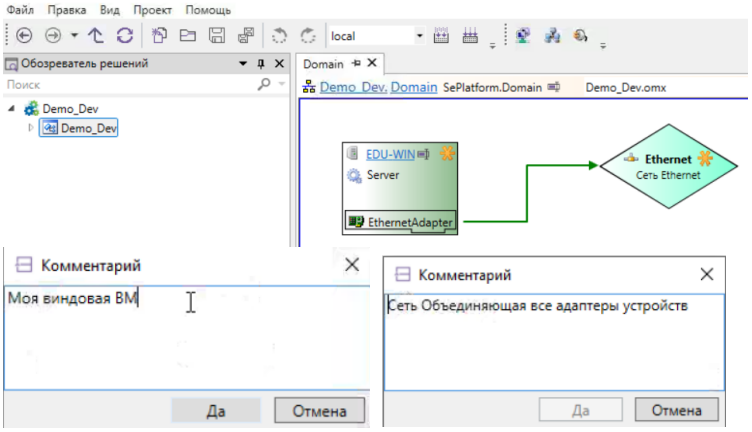
Чтобы добавить комментарий к элементу, щёлкните на него ПКМ → Комментарий. Задайте комментарий элементу ***Домен***.



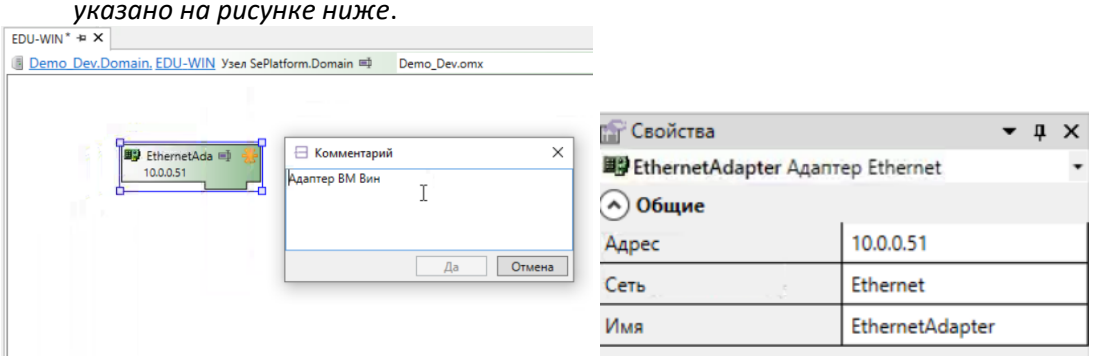


2.5 Перейдите внутрь элемента ***Domain.*** Здесь расположены элементы ***Сеть Ethernet*** и ***Узел SePlatform.Domain***, который необходим для описания исполняющих компонентов и серверов ввода–вывода, соответствующих серверному компьютеру. Задайте комментарии для ***Сети*** и ***Узла SePlatform.Domain.***

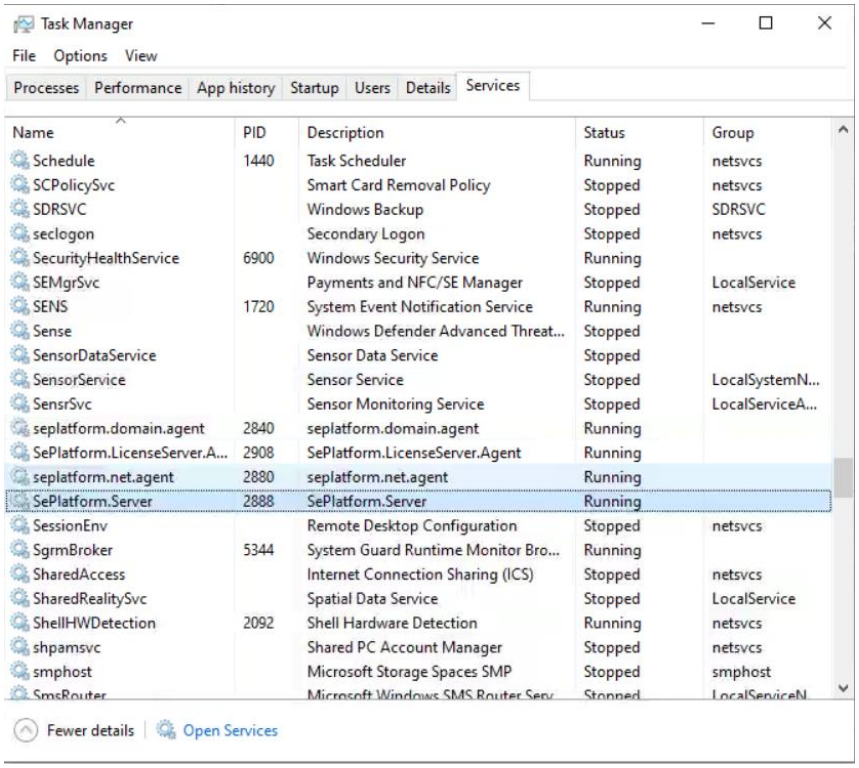
Укажите в качестве имени ***Узла SePlatform.Domain*** (в панели свойств) имя своей виртуальной машины (вместо host), на которой Вы сейчас находитесь (имя машины можно посмотреть через Пуск → Параметры → Система → О программе → Имя устройства).



2.6 Перейдите внутрь ***Узла SePlatform.Domain*** . Здесь в свойстве «Адрес» у ***Ethernet Adapter*** введите IP-адрес машины, на которой Вы разрабатываете проект (IP-адрес машины можно посмотреть через Пуск → Параметры → Сеть и интернет → Свойства, в поиске меню «Пуск» ввести команду CMD, далее в командной строке ввести ipconfig). Задайте комментарий ***Ethernet Adapter*** *как указано на рисунке ниже*.

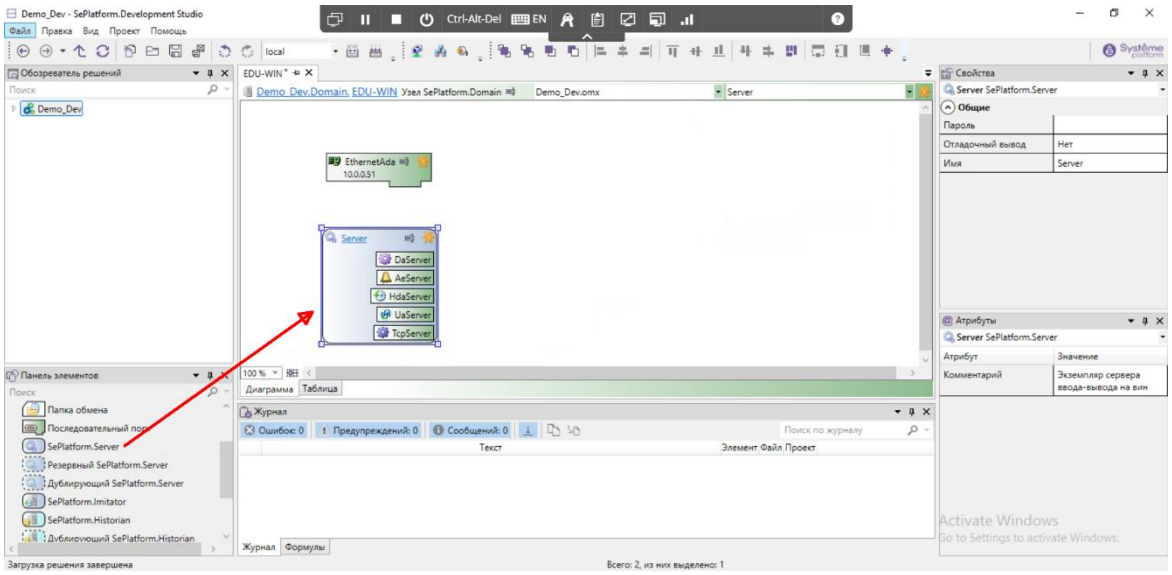


2.7 Так как *SePlatformServer* уже установлен (его можно найти в списке служб), можно начинать его описывать.



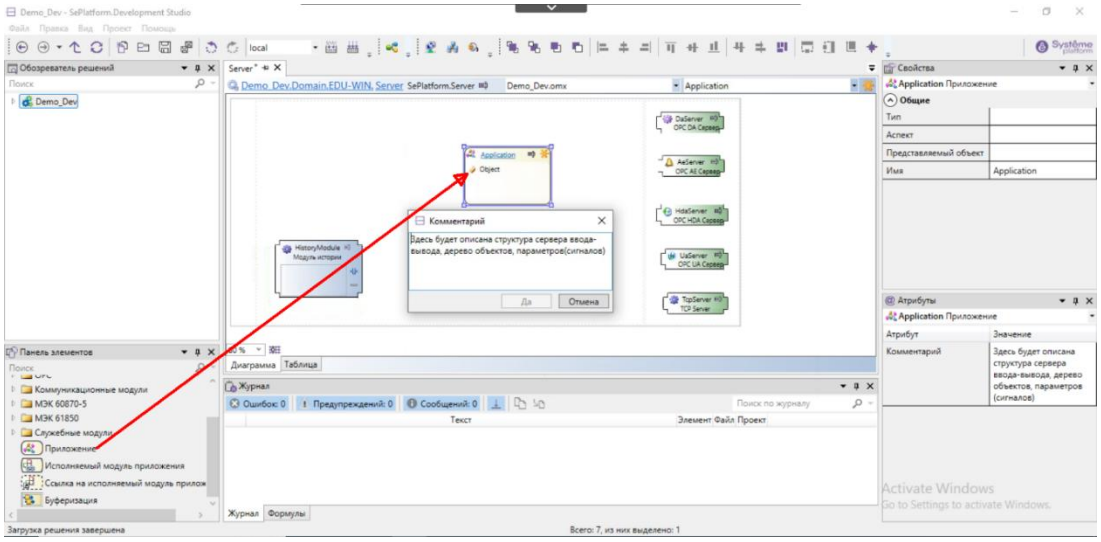
2.8 Внутрь ***Узла SePlatform.Domain*** learning 00 перетяните из панели элементов ***SePlatform.Server***,

2.9 Задайте ему комментарий: *«В данном компоненте будет описан экземпляр сервера ввода-вывода, на который в дальнейшем будет залита конфигурация».*

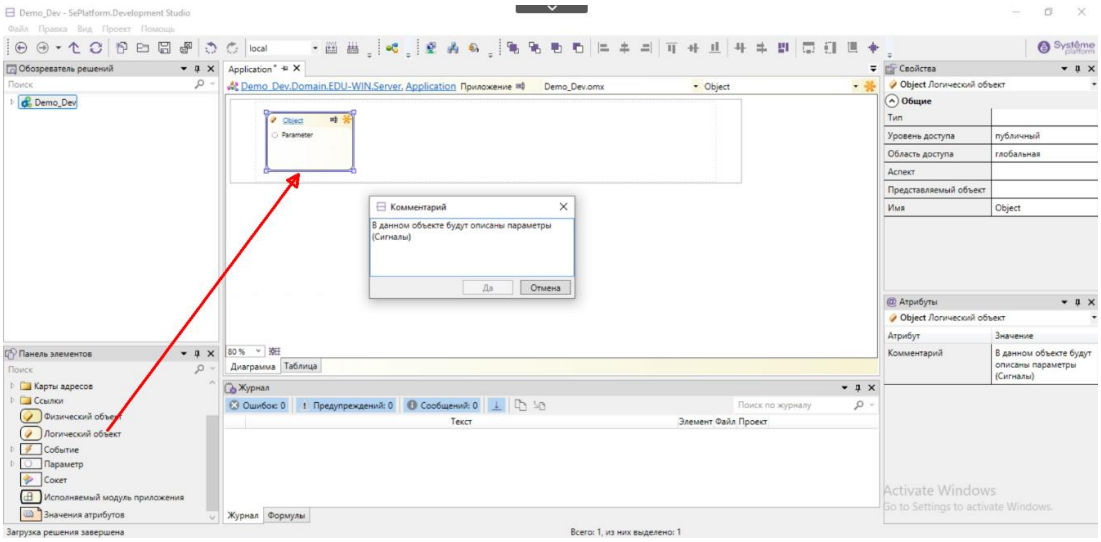


По умолчанию при добавлении Сервера ввода-вывода в конфигурации уже присутствуют модули OPC DA, AE, HAD, UA, TCP Server и модуль истории. Для начала будем использовать модуль OPC DA Server для просмотра оперативных значений сигнала.

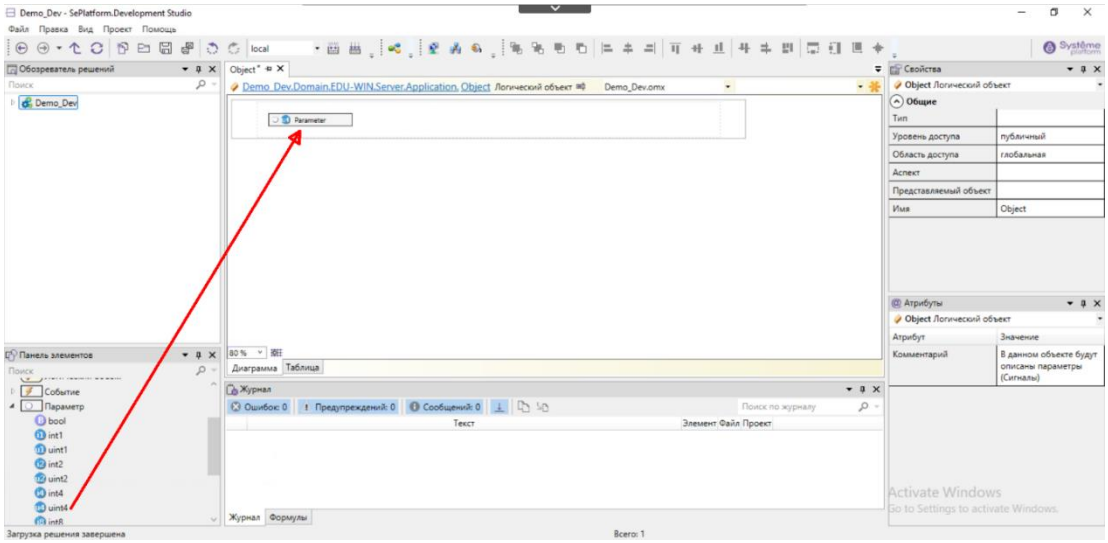
2.10 Перейдите внутрь ***SePlatform.Server.*** Чтобы описать, что будет происходить в этом экземпляре сервера, добавьте элемент ***Приложение*** (из панели элементов). Здесь описывается дерево объектов и параметров, которое будет обслуживаться данным сервером. Задайте ему комментарий как показано на рисунке ниже.



2.11 Перейдите в элемент ***Application*** и перетяните сюда из панели элементов ***Логический объект*** и так же задайте ему комментарий.



2.12 Внутрь этого ***логического объекта*** добавьте ***параметр*** типа Uint4 из панели элементов.



Параметры – сигналы, которые будут в конфигурации *SePlatform.Server*. По отношению к другим компонентам (контроллерам, другим серверам), параметры характеризуются свойством Направление. ***Входные параметры*** влияют на состояние объекта, отправляют управляющие воздействия (команды, отправляемые на контроллер). ***Выходные параметры*** характеризуют состояние объекта (состояние, текущие значения, полученные от контроллера).

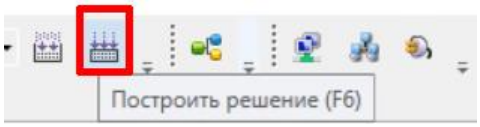
Простой проект готов, осталось его построить и развернуть (применить конфигурацию к определённому экземпляру сервера).

Настройка развёртывания проекта

Перед тем, как применить конфигурацию на определённый экземпляр *SePlatform.Server*, необходимо настроить *SePlatform.Domain*. Подсистема развёртывания состоит из двух служб: **SePlatform.Net.Agent** и **SePlatform.Domain.Agent**. В ходе дальнейшего усложнения проекта в данной методичке будет подробное изложение настройки определённых XML-файлов, относящихся к службам **SePlatform.Net.Agent** и **SePlatform.Domain.Agent**. На данном этапе разработки учебного проекта для развёртывания достаточно настроек по умолчанию.

Построение и развёртывание проекта

При построении решения *SePlatform.DevStudio* строит связи согласно информационной модели обмена данными между элементами в проекте. По описанным связям создаётся конфигурация, которая с помощью подсистемы развёртывания будет применена к нужному экземпляру сервера. Для развёртывания конфигурации необходимо сначала **Построить решение**, затем **Перейти к развёртыванию** для применения конфигурации к нужному экземпляру.

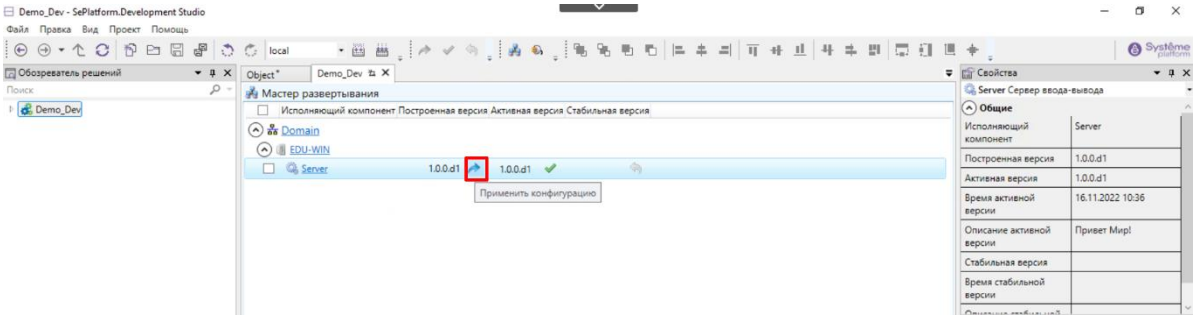


2.13 Нажмите на кнопку **Построить решение**. На данном этапе *SePlatform.DevStudio* компилирует решение и строит связи между компонентами для построения конечных конфигураций *SePlatform.Server*.

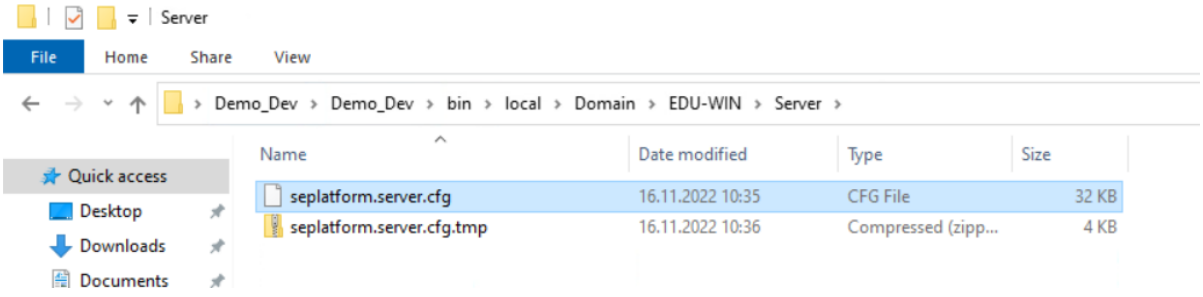
2.14 Нажмите на кнопку **Перейти к развёртыванию**. Откроется Мастер развёртывания, в

котором отображаются сконфигурированные экземпляры *SePlatform.Server* в проекте

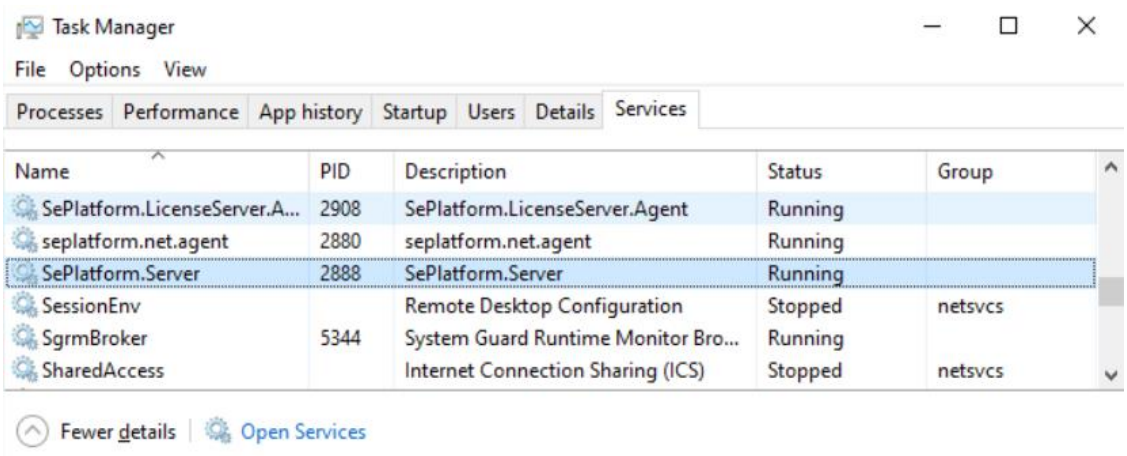
2.15 Нажмите на кнопку **Применить конфигурацию**.



2.16 Во время применения конфигурации подсистема развёртывания останавливает службу нужного экземпляра SePlatform.Server, подкладывает новую готовую конфигурацию для данного экземпляра в папку проекта.



и запускает службу



В результате получается запущенный экземпляр SePlatform.Server на локальной машине с новой готовой конфигурацией, которая описана в проекте. Время применения конфигурации и её версию можно увидеть в Мастере развёртывания в столбце Активная версия.

3. Знакомство и работа с отладочными инструментами SePlatform.Tools

Для отладки и диагностики компонентов *SePlatform Platform* используются набор инструментов *SePlatform.Tools*, в состав которого входят приложения:

- *SePlatform.OpcExplorer* – инструмент для работы с OPC серверами по протоколам DA, AE, HDA, UA, который позволяет:
  - Просматривать адресное пространство источников данных;
  - Просматривать, изменять, генерировать значения сигналов;
  - Просматривать и квитировать события и тревоги;
  - Просматривать изменения сигналов на графике;
  - Запрашивать исторические данные;
- *SePlatform.EventLogViewer* – инструмент для удобного просмотра системных журналов для диагностики работоспособности системы, который позволяет:
  - Просматривать события системных журналов на локальном и удалённом компьютерах;
  - Импортировать и экспортировать системные журналы в файл;
  - Фильтровать и искать события.

Работа с SePlatform.OpcExplorer

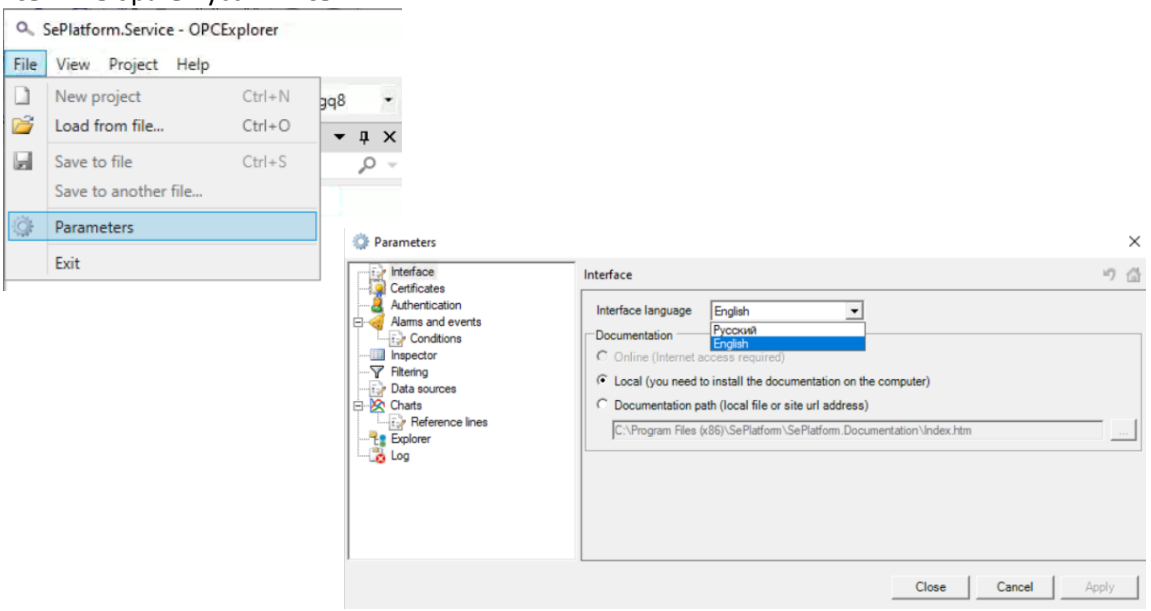
Для просмотра значений сигналов можно использовать *SePlatform.OpcExplorer.*

Работа с сигналами. Подключение к OPC DA Server

SePlatform.OpcExplorer позволяет подключаться к различным OPC (DA, AE, UA, HDA) источникам, просматривать, изменять значения.

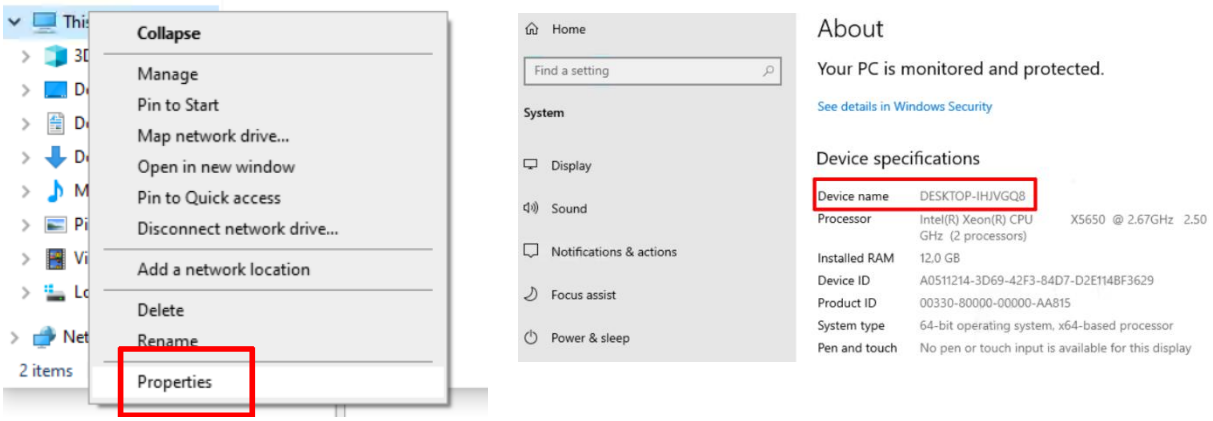
3.1 Откройте *SePlatform.OpcExplorer* (Пуск → SePlatform → SePlatform.OpcExplorer).

Подсказка, в этом инструменте также можно сменить язык интерфейса на Русский нужно выбрать Файл → Параметры после чего выбрать параметр интерфейс и в выпадающем списке «Сменить язык» выбрать Русский язык

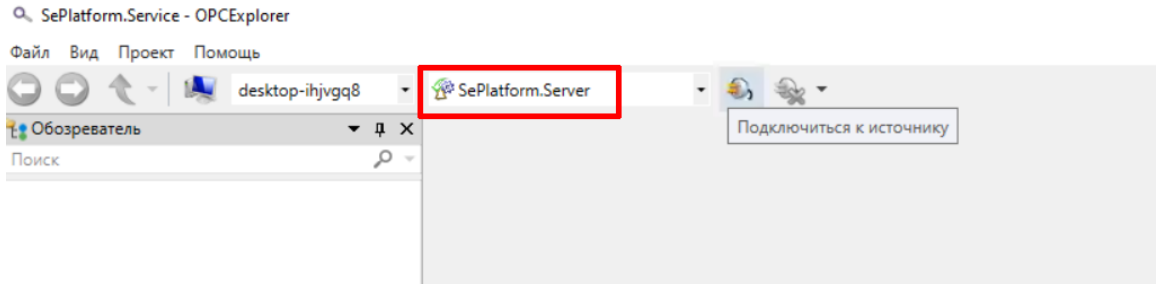


3.2 Выберите в поле «Компьютер» имя своей машины, нажмите клавишу Enter.

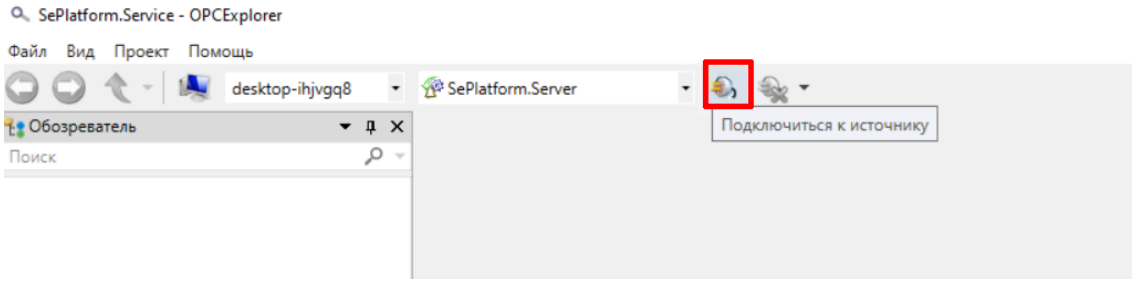
Подсказка – имя ВМ можно посмотреть в свойствах ПК щелкнув ПКМ по иконке «Этот Компьютер»



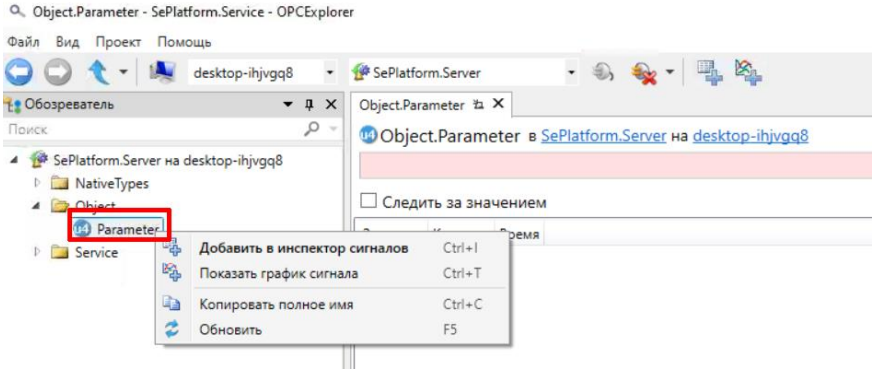
3.3 В поле «Источник данных» выберите Server.



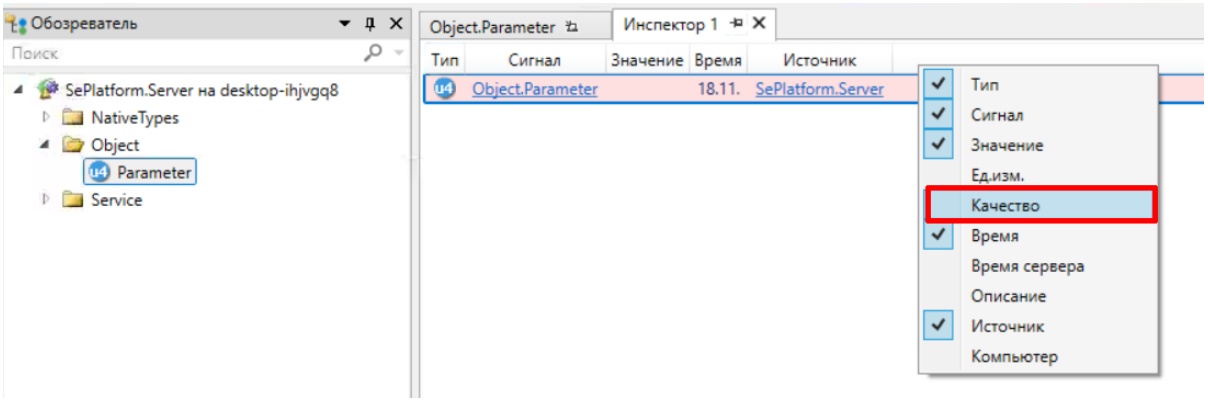
3.4 Нажмите на кнопку **Подключиться к источнику**.



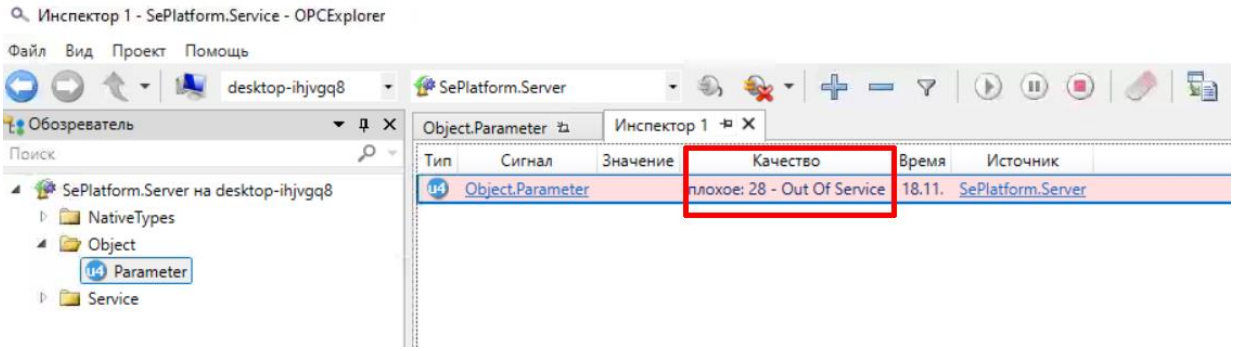
3.5 Раскройте дерево OPC источника, к которому произведено подключение, и выберете сигнал, имеющийся в конфигурации. Добавьте сигнал в инспектор сигналов (в дереве ПКМ по сигналу → Добавить в инспектор сигналов).



Подсказка: чтобы добавить поля просмотра нужно кликнуть ПКМ по типу поля и в появившемся окне выбрать необходимый параметр

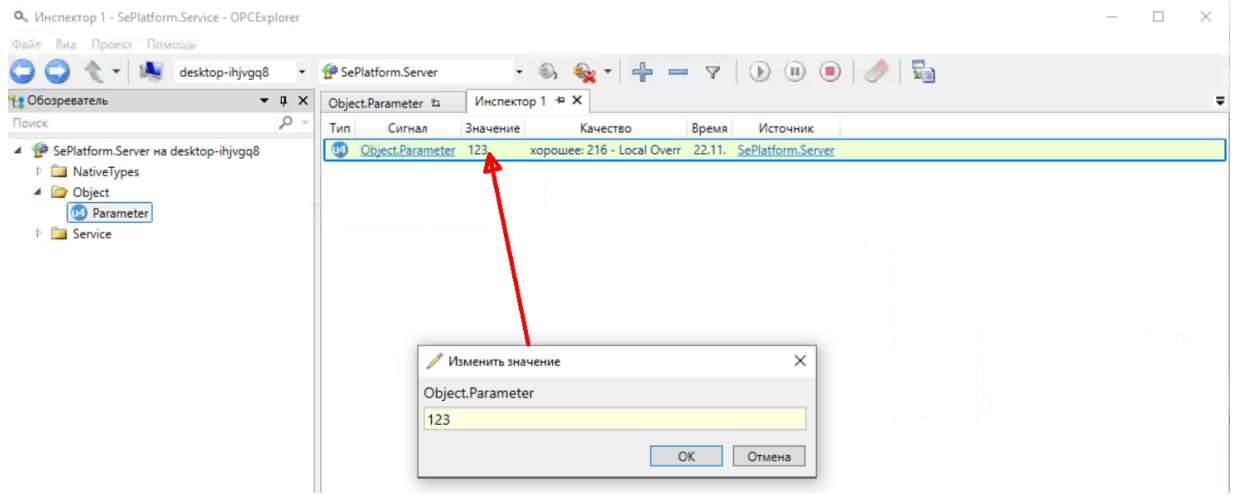


Так как данный сигнал не получает данные с коммуникационного модуля, по умолчанию у него плохое качество (код 28) и пустое значение.

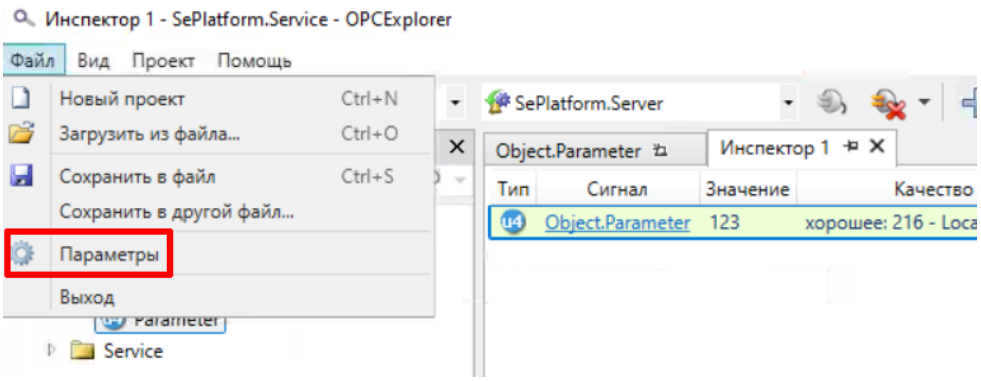


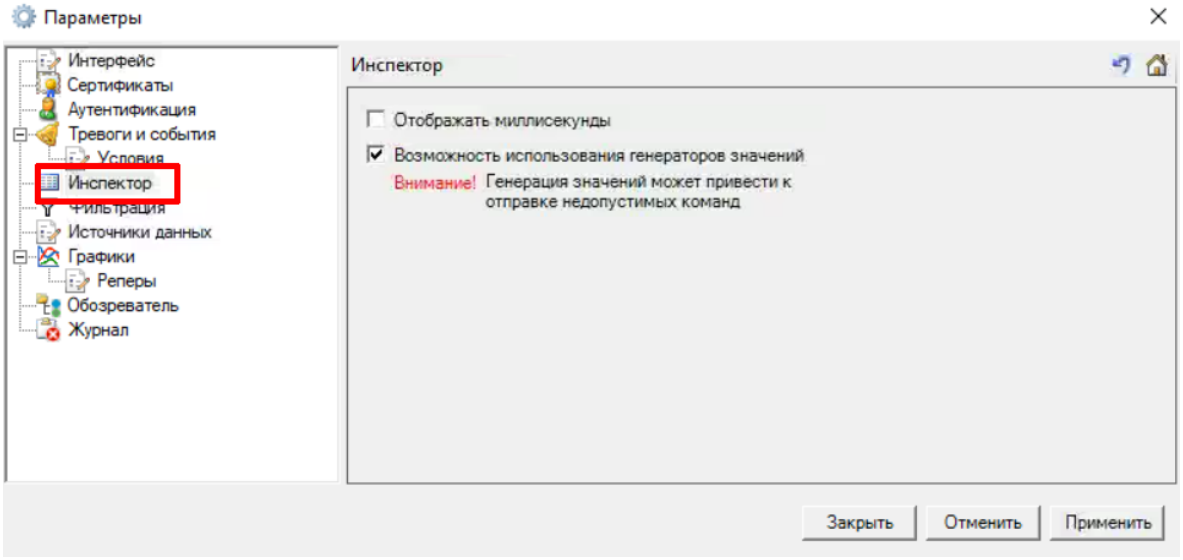
С помощью инспектора сигналов можно просматривать и изменять текущие значения сигналов.

3.6 Для изменения значения сигнала дважды щёлкните мышкой в графе «Значение» данного сигнала в инспекторе. В появившемся окне установите необходимое значение.



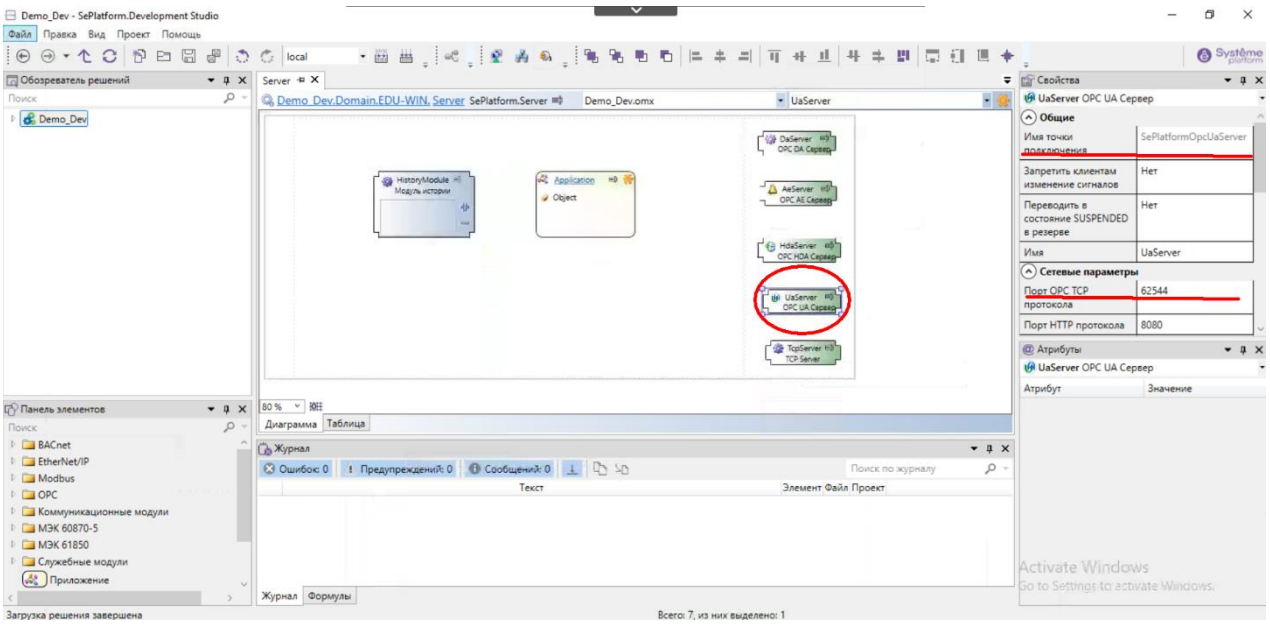
3.7 Для того, чтобы не вводить значения вручную, можно добавить генератор значений. Для его работы необходимо включить его в настройках (Файл → Параметры → Инспектор → Возможность использования генераторов значений → Сохранить → Применить). После этого снова откройте редактор значения сигнала, установите флаг Генерировать значения, нажмите кнопку **Добавить**, выберите Случайный генератор и нажмите **ОК**. Теперь у сигнала будут генерироваться случайные значения.





Подключение к OPC UA Server

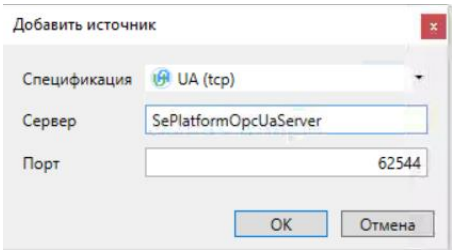
В *DevStudio* внутри сервера есть модуль OPC UA Server. Этот модуль является кроссплатформенным: с его помощью можно подключиться не только к серверу, установленному на ОС Windows, но и к серверу на ОС Linux.



3.8 Для подключения к OPC UA Server перейдите в *SePlatform.OpcExplorer,* введите IP-адрес своей машины, нажмите клавишу Enter, нажмите на **кнопку с монитором** (Запрос проверки связи ICMP), расположенную слева.

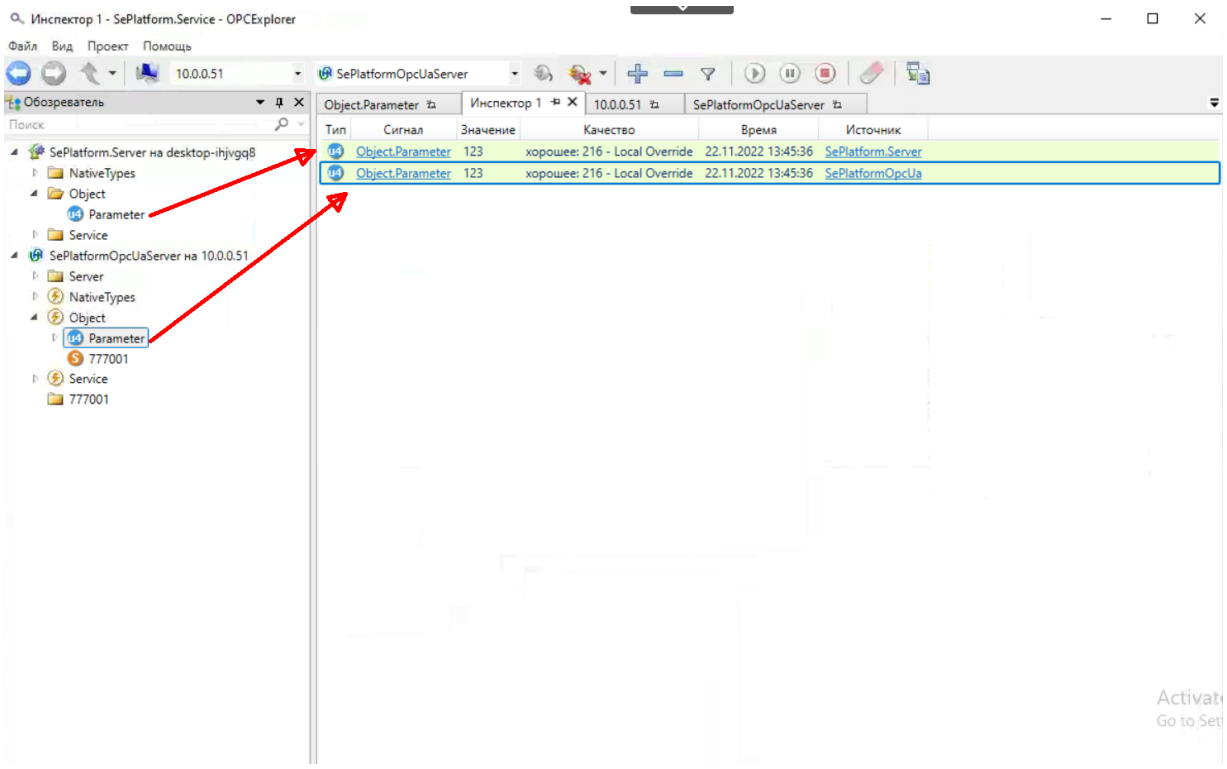
3.9 Нажмите на кнопку с изображением знака  (Добавить источник).

3.10 В строке «Спецификация» выберите UA (tcp), в графу «Сервер» впишите Имя точки подключения, указанную в *DevStudio (см. рисунок выше),* сравните порт из *OpcExplorer* с портом OPC TCP протокола из *DevStudio.* Нажмите ОК.



3.11 В *OpcExplorer* в поле Источник данных выберите добавленный сервер и подключитесь к нему.

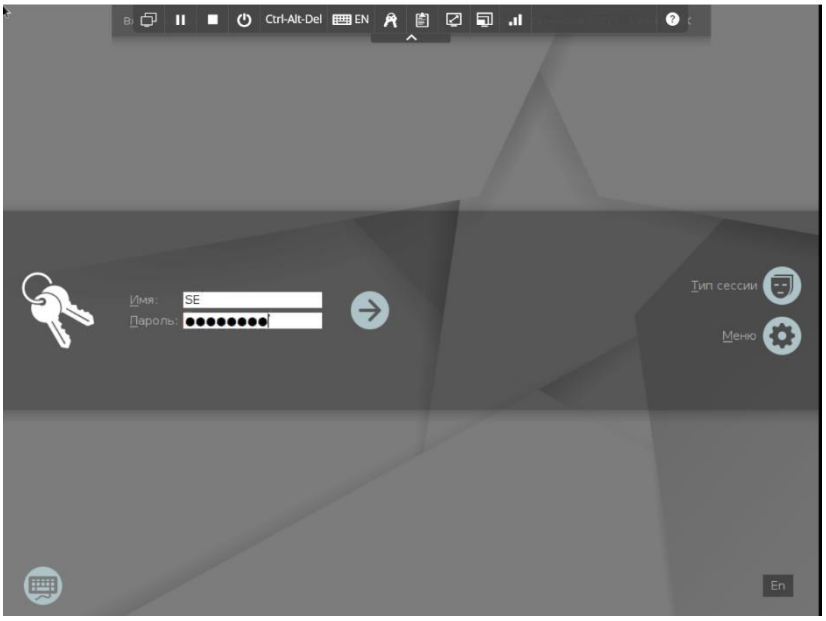
3.12 Добавьте **Параметр** из логического типа Object в Инспектор сигналов.



Теперь Вы подключены к серверу Windows по OPC UA. Можно как считывать значения, так и записывать. При этом значение переменной уже было задано по другому протоколу и можно увидеть, что мы получаем идентичную информацию по всем протоколам.

4. Установка и настройка серверной части на Linux. Создание узла для экземпляра SePlatform.Server на Linux в SePlatform.DevStudio.

Для выполнения работ вам потребуется подключиться ко второй виртуальной машине с уже предустановленной системой Linux.

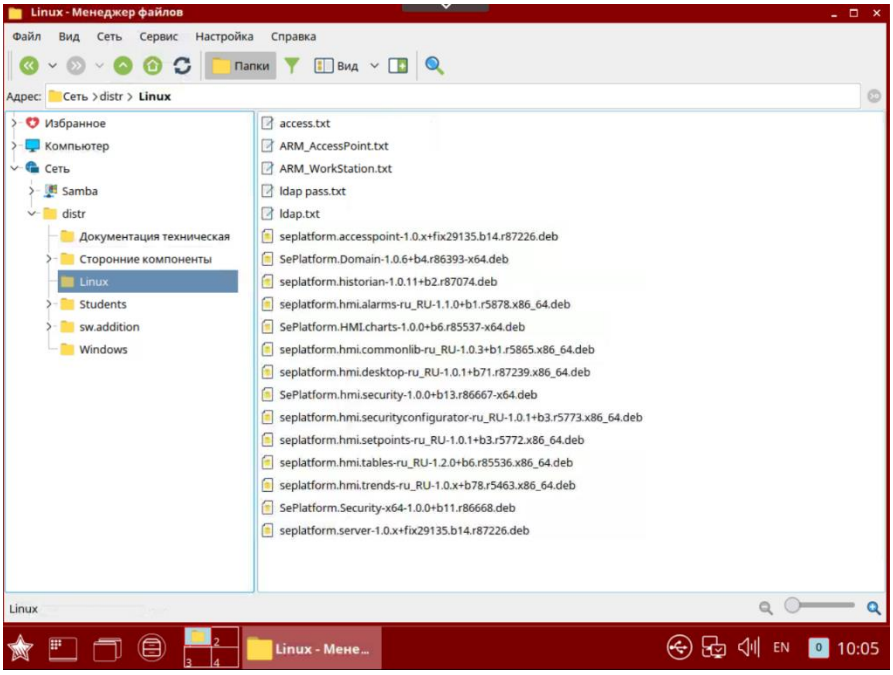


Имя пользователя – SE
Пароль – 12345678

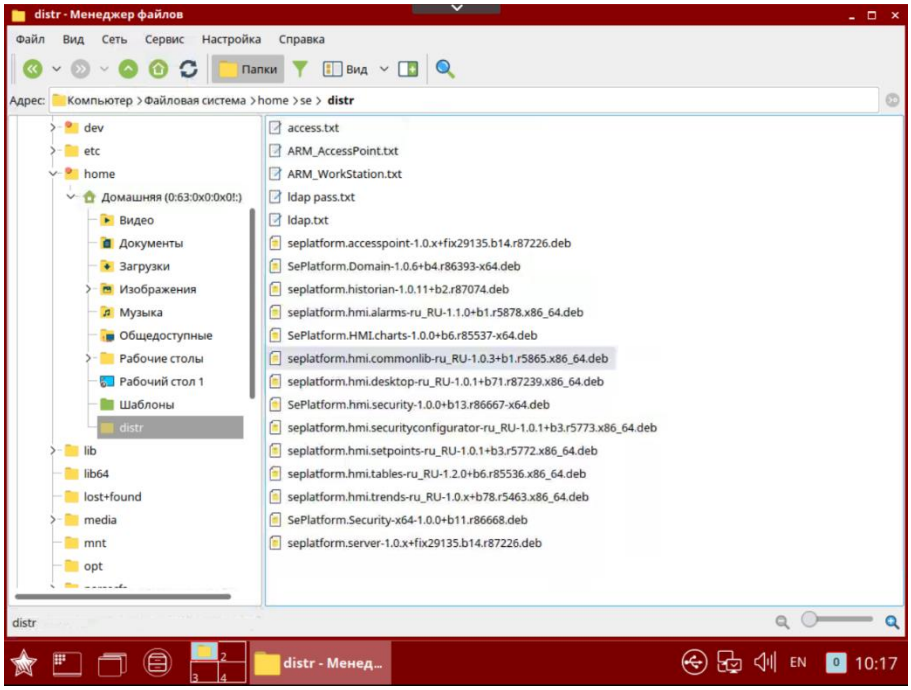
Для начальной работы нужно установить *SePlatform.Server* и *SePlatform.Domain*. Дистрибутивы находятся в сетевой папке машины Linux.

4.1 23333333333Откройте менеджер файлов.

4.2 В открывшемся окне выберите Сеть → Distr → Linux . В данной папке предварительно собраны все дистрибутивы

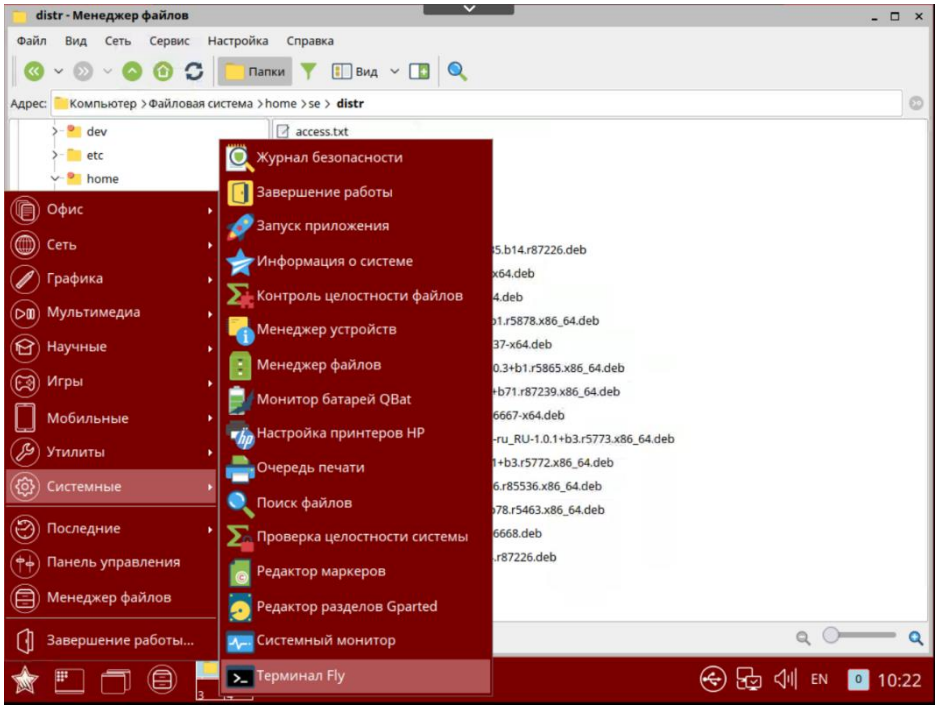


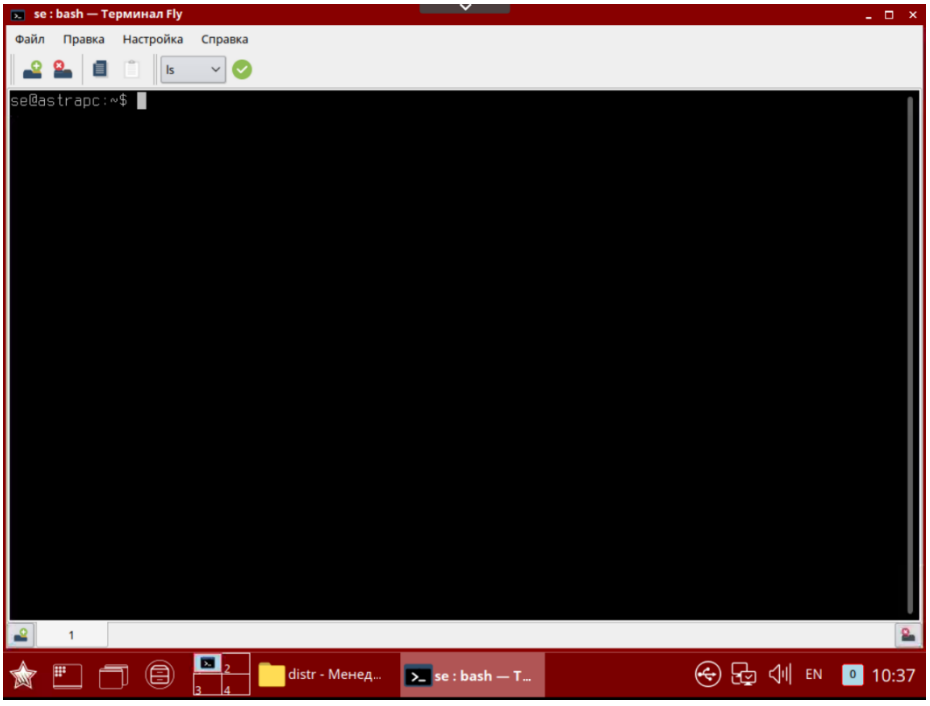
4.3 Чтобы работать с дистрибутивами их необходимо скопировать на вашу ВМ в любой каталог, в который позволяет система. Удобно работать через папку /home/se/distr



Документацию по работе Systeme Platform можно посмотреть в разделе «Работа с ОС Linux»

4.4 Чтобы начать работу с терминалом Linux нужно запустить терминал Fly (Пуск → Системные → терминал Fly)



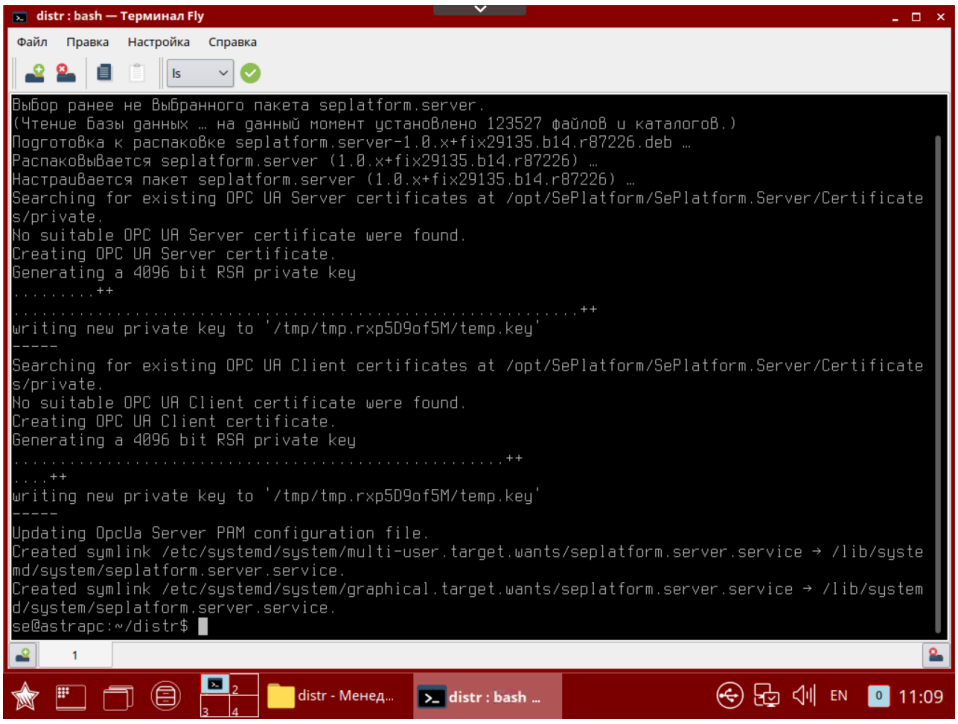


4.5 Далее необходимо открыть папку с дистрибутивами через терминал. Для этого нужно ввести cd /home/se/distr

4.6 Сначала необходимо установить *SePlatform.Server* на ОС Linux описаны в документации: SePlatform.Server → SePlatform.Server → Подготовка к работе → Стандартная установка ОС Linux.

Находясь в папке с установочным пакетом, введите команду **sudo dpkg -i SePlatform.server-*******.deb**

Подсказка: чтобы избежать ошибок при вводе команд Linux можно использовать функцию «Автодополнение» для этого достаточно ввести часть команды, а затем нажать клавишу «tab» на клавиатуре



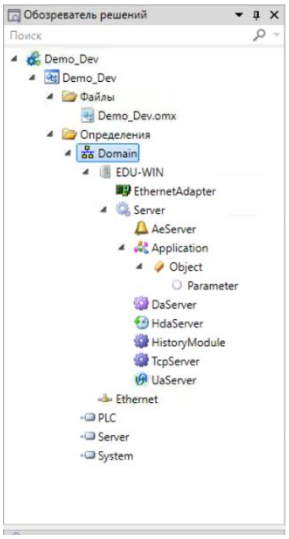
нажмите клавишу Enter.

Для установки *SePlatform.Domain*, находясь в папке с установочным пакетом, введите команду **sudo dpkg -i SePlatform.domain-********.deb** и нажмите клавишу Enter. (повторить П 4.5 и 4.6)
После установки *SePlatform.Server* и *SePlatform.Domain,* вернитесь в *DevStudio* для описания Linux машины в среде исполнения.

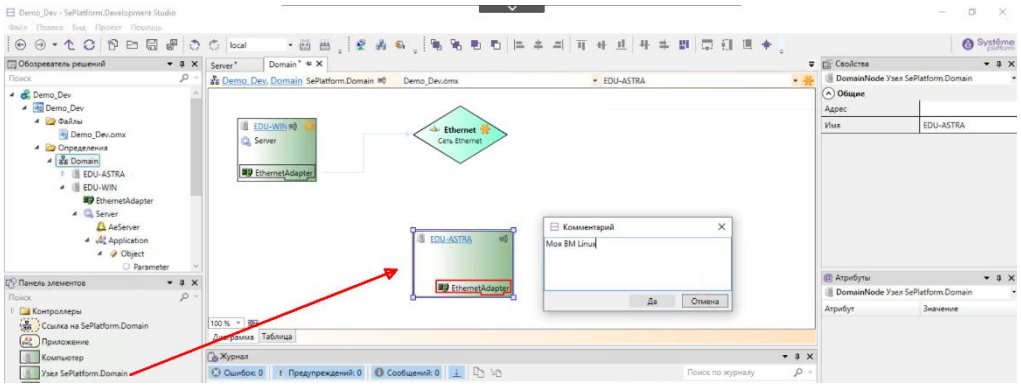
4.7 Описание конфигурации *SePlatform.Server* на Linux машине в DevStudio

Для описания ещё одной машины в проекте необходимо перейти в элемент ***Domain*** (здесь описываются все машины, участвующие в проекте).

Перейдите в элемент ***Domain*** через Обозреватель решений.

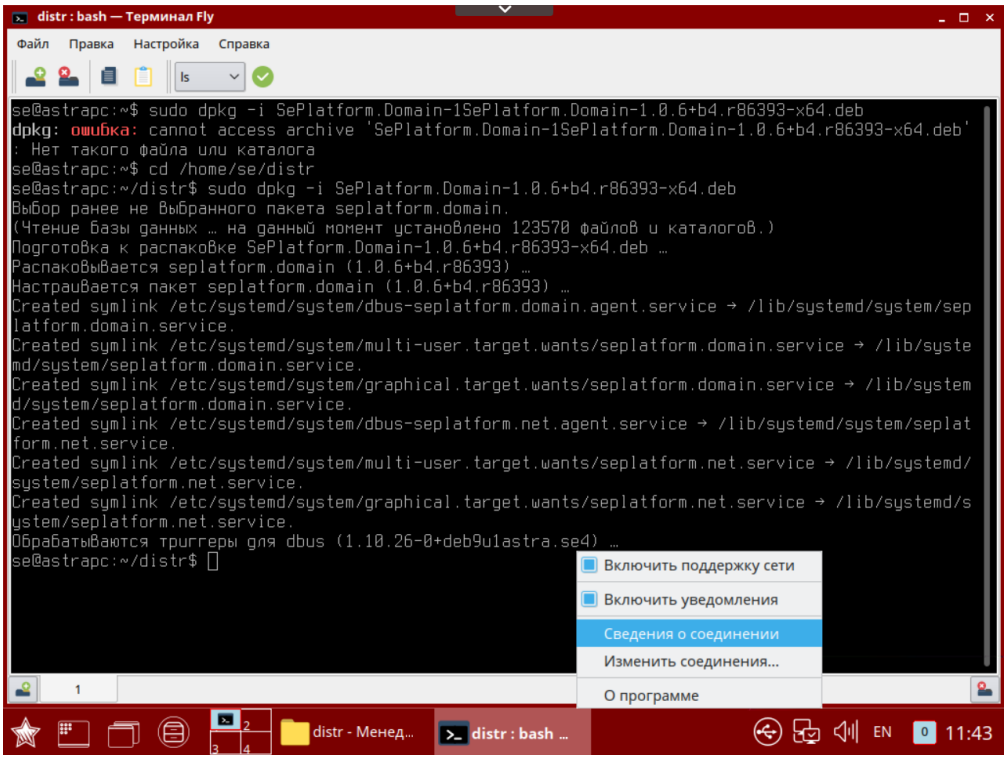


4.8 Перетяните на рабочую область ***Узел SePlatform.Domain*** из панели элементов, задайте ему имя, совпадающее с именем машины с ОС Linux, адрес (любой), добавьте комментарий как на рисунке ниже (ПКМ по *Узел SePlatform.Domain → Комментарий).*

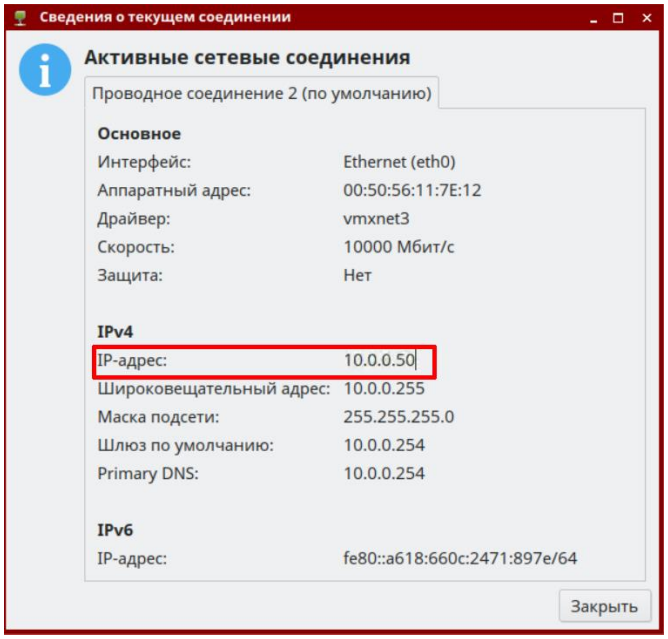


4.9 Перейдите внутрь созданного ***Узла SePlatform.Domain*** для Linux машины (в дальнейшем astra), выделите ***EthernetAdapter***, в свойстве Адрес укажите IP-адрес Linux машины

Подсказка: чтобы узнать IP адрес ВМ Linux можно переключившись на нее в панели управления кликнуть ПКМ по иконке «сетевое соединение ethernet» и выбрать «сетевое соединение»

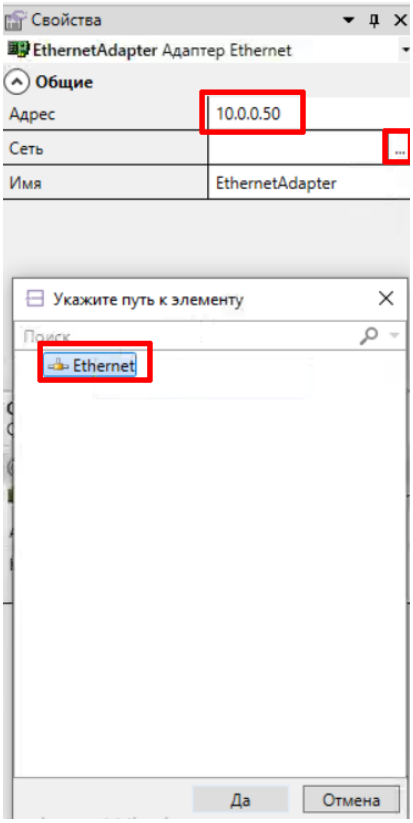


После чего в отобразившемся диалоговом окне можно будет посмотреть IP адрес LinuxBM

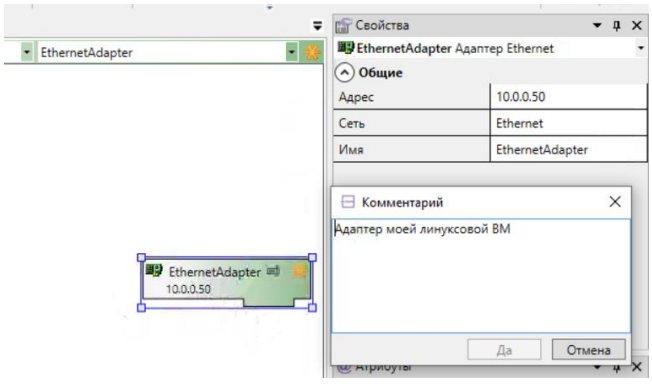


в свойстве Сеть при помощи двойного нажатия кнопки мыши выберите Ethernet,

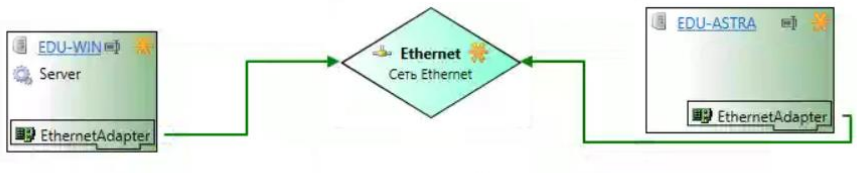
Подсказка: в качестве альтернативы вместо навигации по меню для того, чтобы узнать IP-адрес можно использовать команду sudo ifconfig в терминале Fly.



добавьте комментарий как на рисунке ниже.

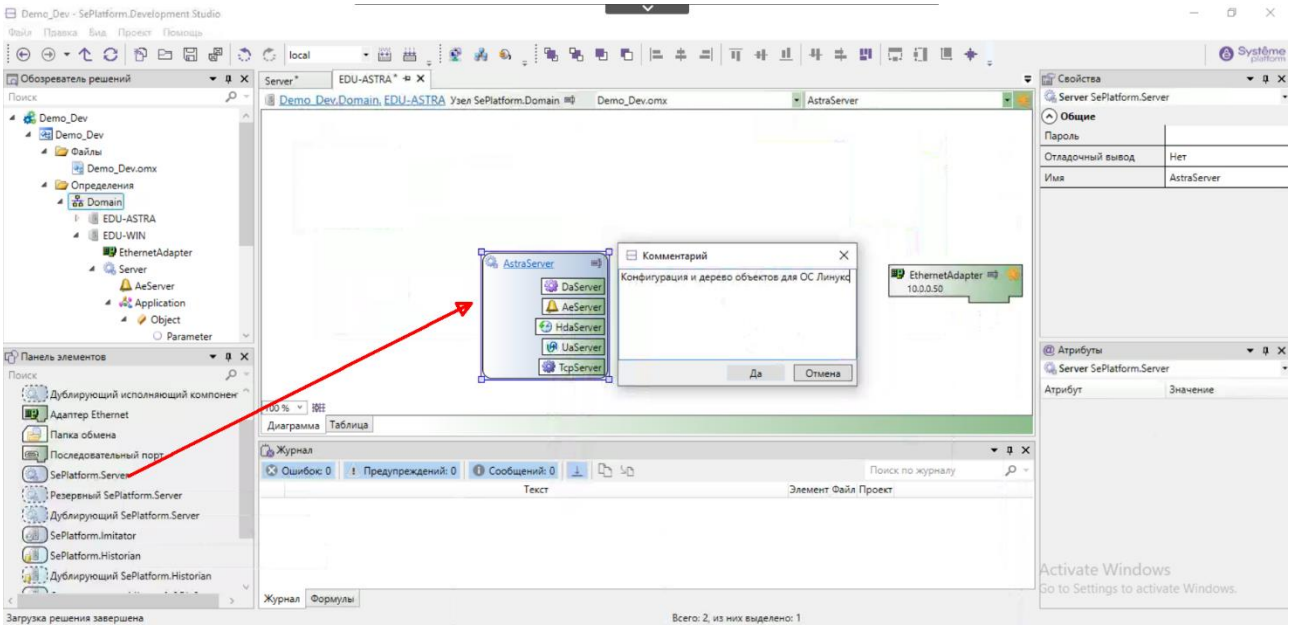


Поднявшись на уровень выше к элементу ***Domain***, можно заметить, что два узла подключены к одной сети, т.е. имеют адаптеры, которые работают через одну общую сеть и связаны между собой. В

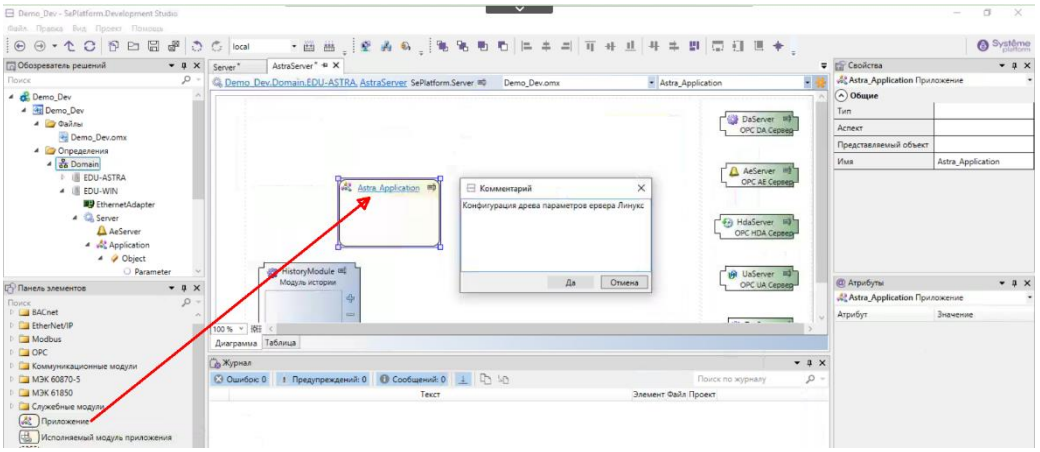


качестве альтернативы можно, зажав ЛКМ провести связь между узлом EDU-Astra и сетью Ethernet, это также добавит связь.

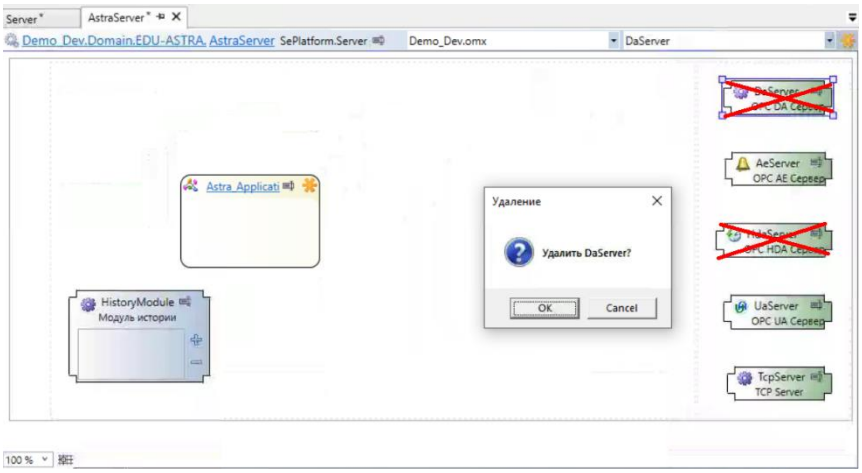
4.10 Перейдите внутрь узла ***astra*** и перетяните на рабочую область ***SePlatform.Server*** из панели элементов, дайте ему имя, отличное от имени сервера внутри виндового узла. Задайте ему комментарий как на рисунке ниже.



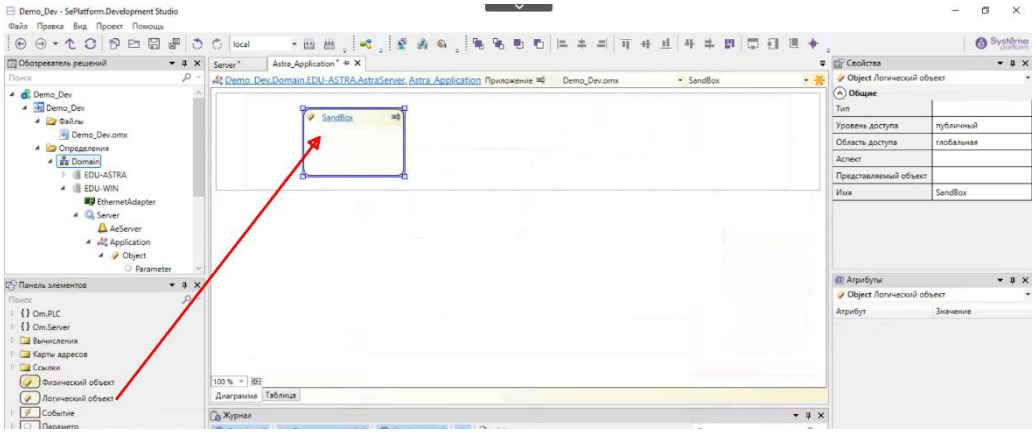
4.11 Перейдите внутрь ***AstraServer***. Перетяните из панели элементов ***Приложение***, дайте ему название и комментарий как на рисунке ниже.



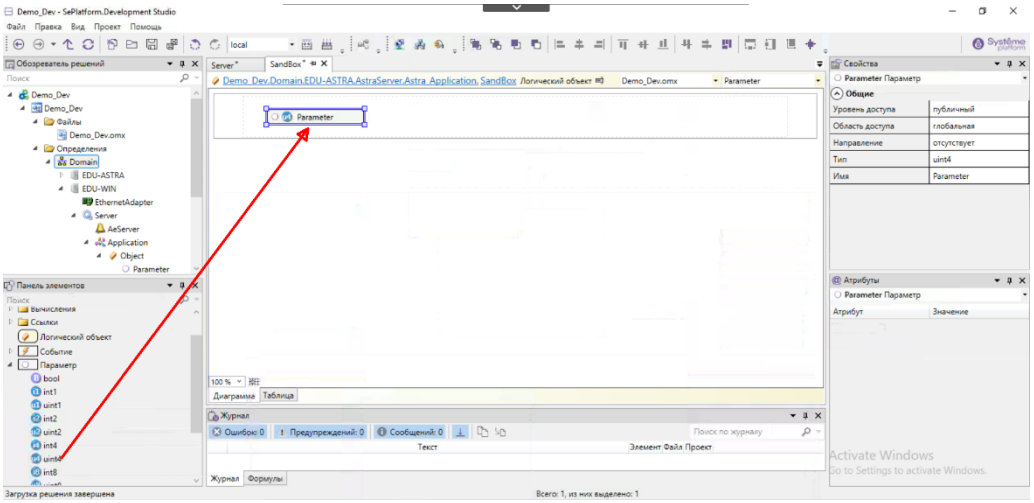
На машинах с ОС Linux не доступны OPC DA и HDA Server, поэтому их можно удалить.



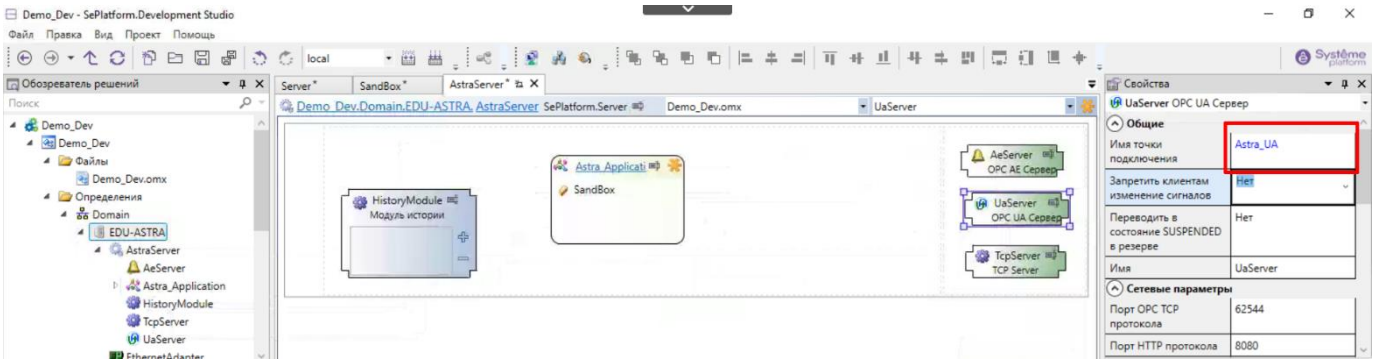
4.12 Перейдите внутрь ***Astra_Application*** и перетяните из Панели элементов ***Логический объект***, назовите его SandBox.



4.13 Перейдите внутрь ***SandBox*** и добавите в этот логический объект ***параметр*** типа uint4.



4.14 На уровне сервера для подключения OPC UA зададим уникальное имя точки подключения **Astra_UA** для удобства дальнейшего отображения.



Подсказка: чтобы посмотреть список серверов в Linux, которые связаны с SePlatform нужно запустить Терминал Fly и ввести команду - sudo systemctl list-units | grep seplatform

Конфигурация *SePlatformServer* на Linux машине готова. Но для того, чтобы развернуть эту конфигурацию, сначала нужно настроить XML-файлы **SePlatform.Domain.Agent** и **SePlatform.Net.Agent** на Wndows и Linux машинах.

5. Конфигурирование SePlatform.Domain

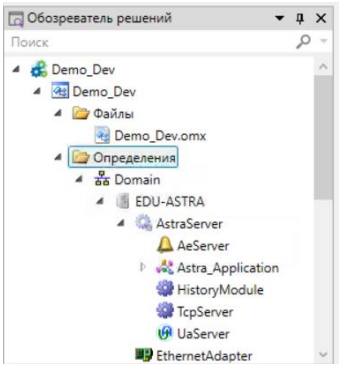
SePlatform.Domain представляет собой набор из двух служб, которые отвечают за то, чтобы развернуть конфигурации на определённой машине: **SePlatform.Net.Agent** и **SePlatform.Domain.Agent**. **SePlatform.Net.Agent** связывает данный компьютер с другими компьютерами в ***сети SePlatform.Net***. *SePlatform.Domain* отвечает за применение конфигурации определённой службы к определённому Systeme Platform серверу на каждой конкретной машине.

Файлы конфигураций находятся в папке:

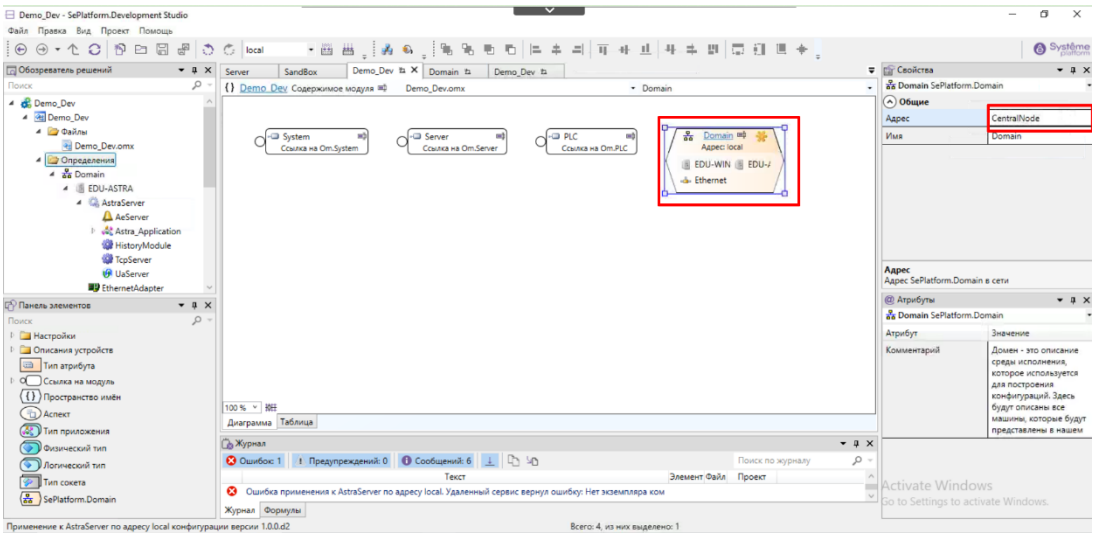
- ОС Windows: C:\Program Files\SePlatform\SePlatform.Domain
- ОС Linux: /opt/SePlatform/SePlatform.Domain

Конфигурирование SePlatform.Domain на ОС Windows

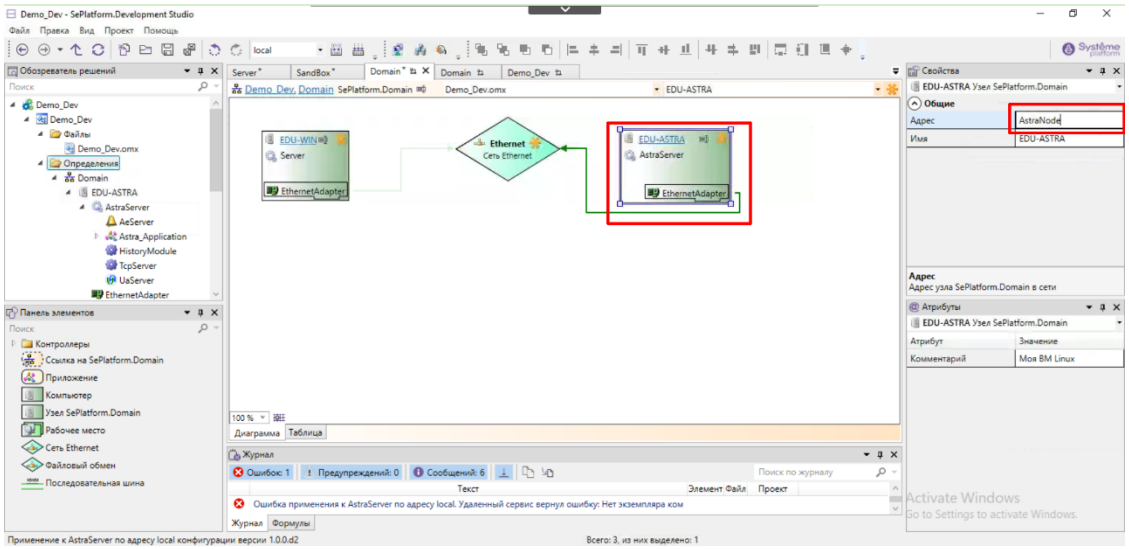
5.1 Сначала перейдите в элемент ***Определения*** при помощи Обозревателя решений.



Выделите элемент ***Domain*** и в свойстве Адрес задайте CentralNode.

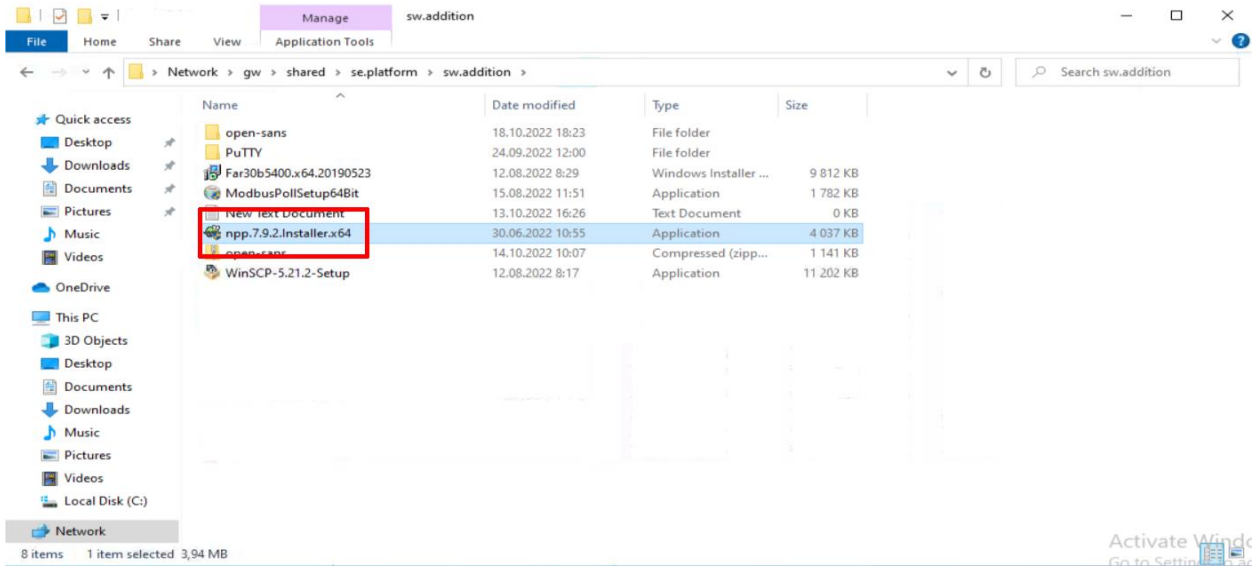


5.2 Задайте Linux машине адрес AstraNode.



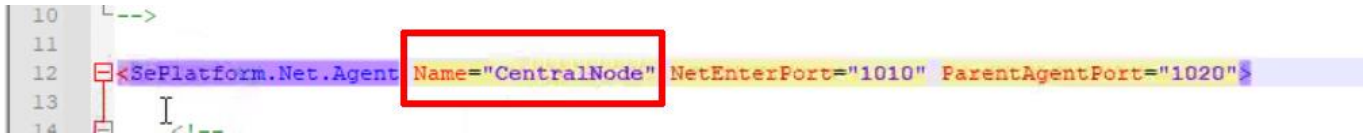
После этого можете переходить к конфигурированию *SePlatform.Domain.*

5.3 Установите Notepad++.Дистрибутив находится в сетевой папке \\gw\shared\se.platform\sw.addition)

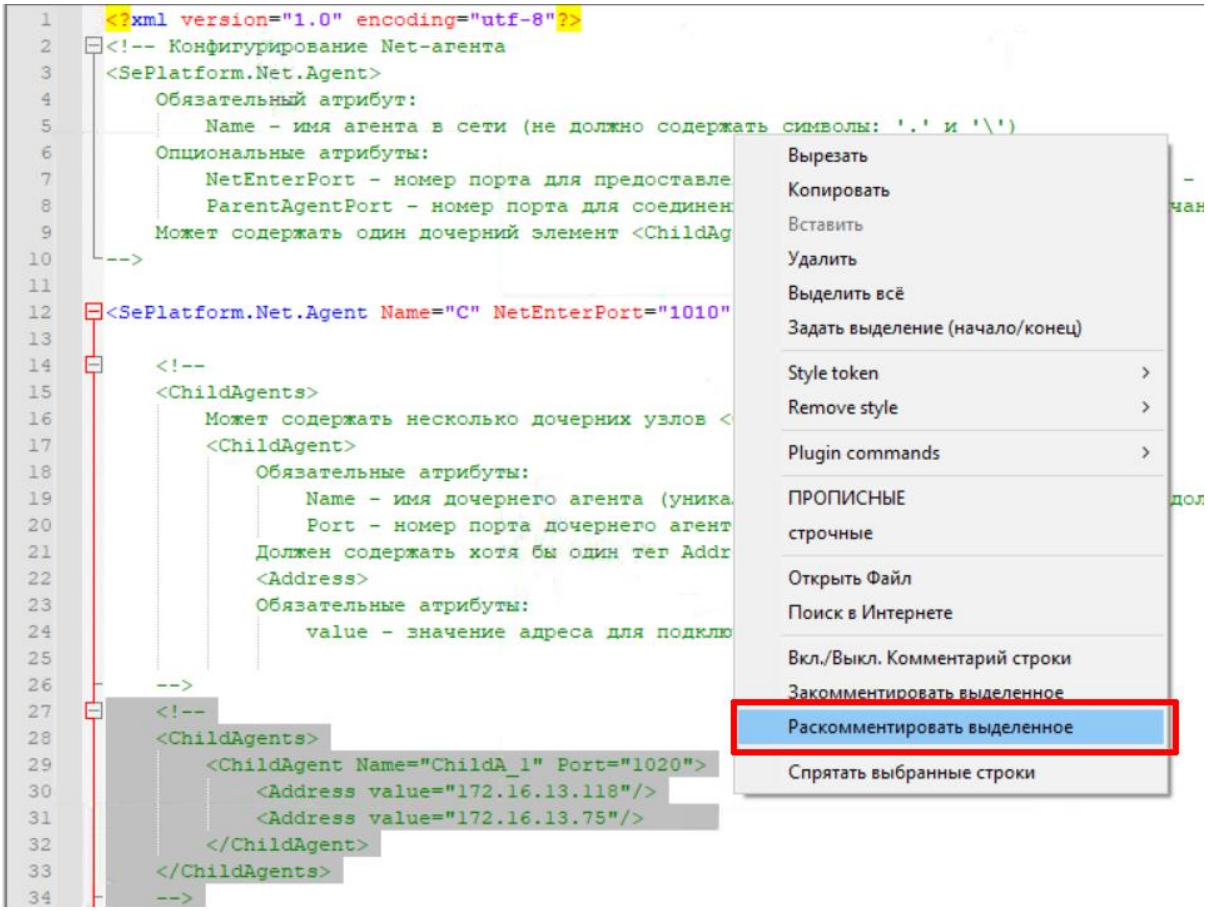


5.4 Перейдите в папку с файлами конфигураций на Wndows машине(C:\Program Files\SePlatform\SePlatform.Domain), откройте файл **SePlatform.net.agent.xml** с помощью Notepad++.

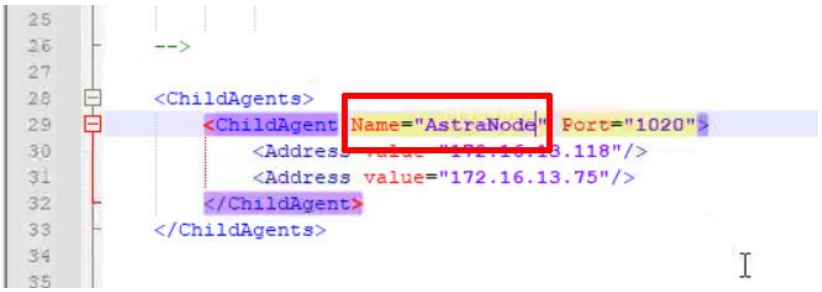
5.5 Найдите строчку с тэгом **SePlatform.Net.Agent**. В атрибуте **Name** необходимо вписать содержимое в свойстве Адрес элемента *Domain* из DevStudio: CentralNode.



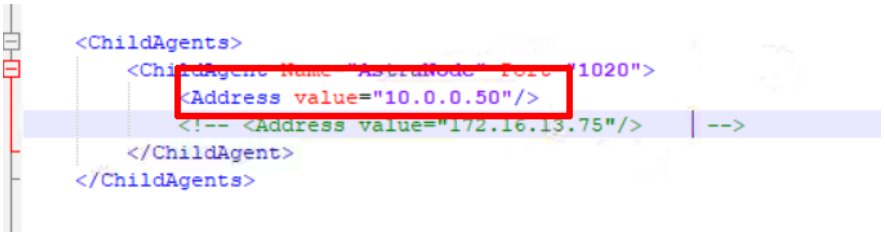
5.6 Уберите комментарий с блока **<ChildAgents>**: выделите с 27 по 34 строчки (нажмите ПКМ и выберите Раскомментировать выделенное). В этом блоке описаны все дочерние узлы. В разрабатываемом проекте главным узлом является виндовая машина, а дочерним – линуксовая.



5.7 В атрибуте **Name** тэга **ChildAgent** введите содержимое свойства Адрес Linux машины,



в атрибуте **Address value** укажите IP-адрес Linux машины. Дополнительно можно закомментировать строчку со вторым адресом, она нам не нужна



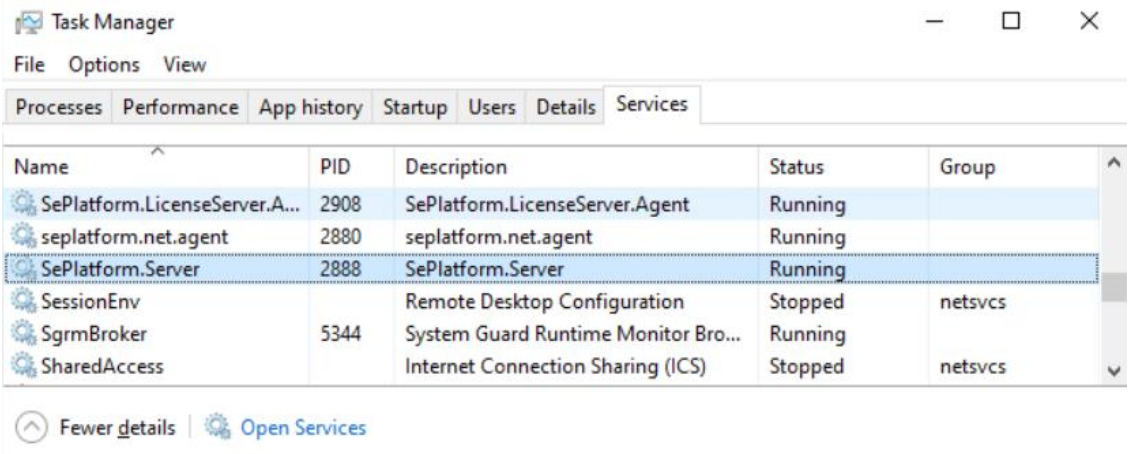
5.8 Сохраните **SePlatform.net.agent.xml**.

Подсказка: Notepad++ сохранит внесенные изменения после подтверждения прав администратора. После того как права были подтверждены нажмите «Сохранить» еще раз

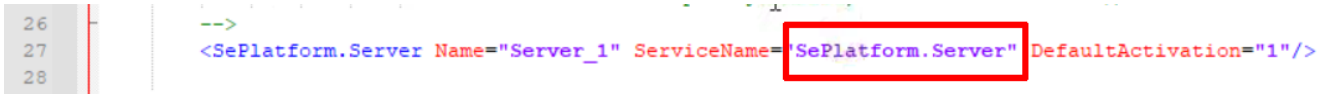
5.9 Откройте файл **SePlatform.domain.agent.xml** с помощью Notepad++.

5.10 Найдите строчку с тэгом **EntryPointNetAgent**. В атрибуте Name необходимо вписать содержимое в свойстве Адрес элемента *Domain* из DevStudio: CentralNode**.**

В тэге **<InstalledComponents>** в атрибуте **ServiceName** описывается имя службы, которая обслуживает исполняющий компонент – элемент *SePlatform.Server* в проекте *DeveloperStudio*. Имя этой службы можно посмотреть в диспетчере задач во вкладке «Службы». Это же имя должно быть прописано в атрибуте **ServiceName** в файле **SePlatform.domain.agent.xml.**



Тэг **Components** внутри **<Server>** описывает установленные компоненты, работающие в рамках роли сервера. Атрибут **Name** должен содержать имя элемента *SePlatform.Server* в проекте в *DeveloperStudio*. То есть служба **SePlatform.Server** обслуживает исполняющий компонент ***Server*** внутри узла, характеризующего виндовую машину.



5.11 В атрибуте **Name** укажите «Server_Central» в качестве Alias

5.12 Далее в коде находится строчка **Installed Name**, которая описывает связку службы с компонентом по имени. Ее название также необходимо изменить на Server_Central

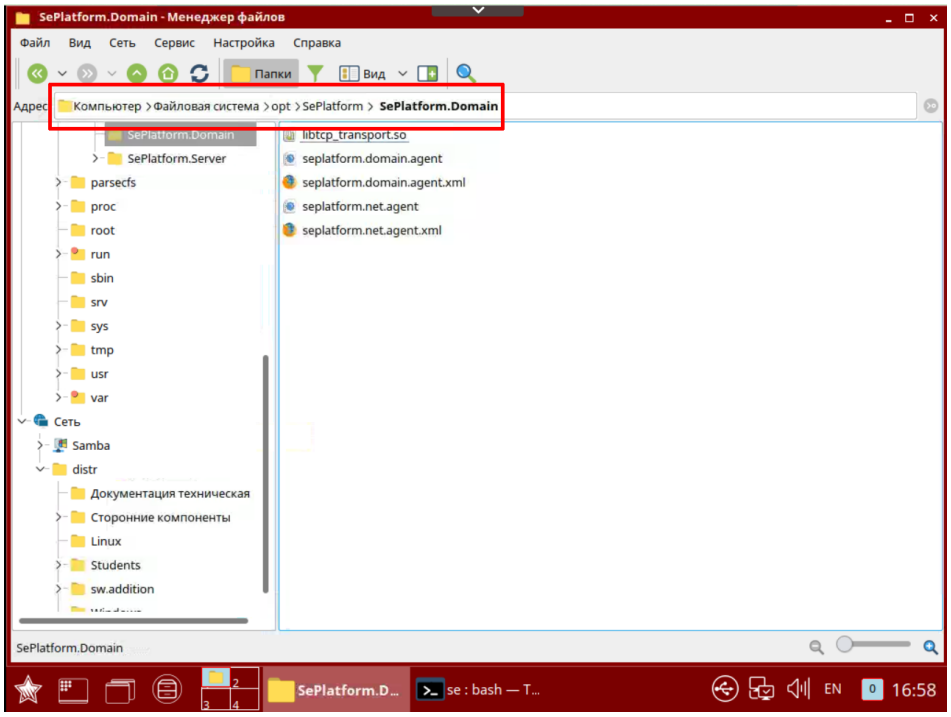
Сохраните **SePlatform.domain.agent.xml.**

5.13 Откройте Диспетчер задач и перейдите на вкладку Службы. Здесь при помощи ПКМ перезапустите службы **SePlatform.Net.Agent** и **SePlatform.Domain.Agent**.

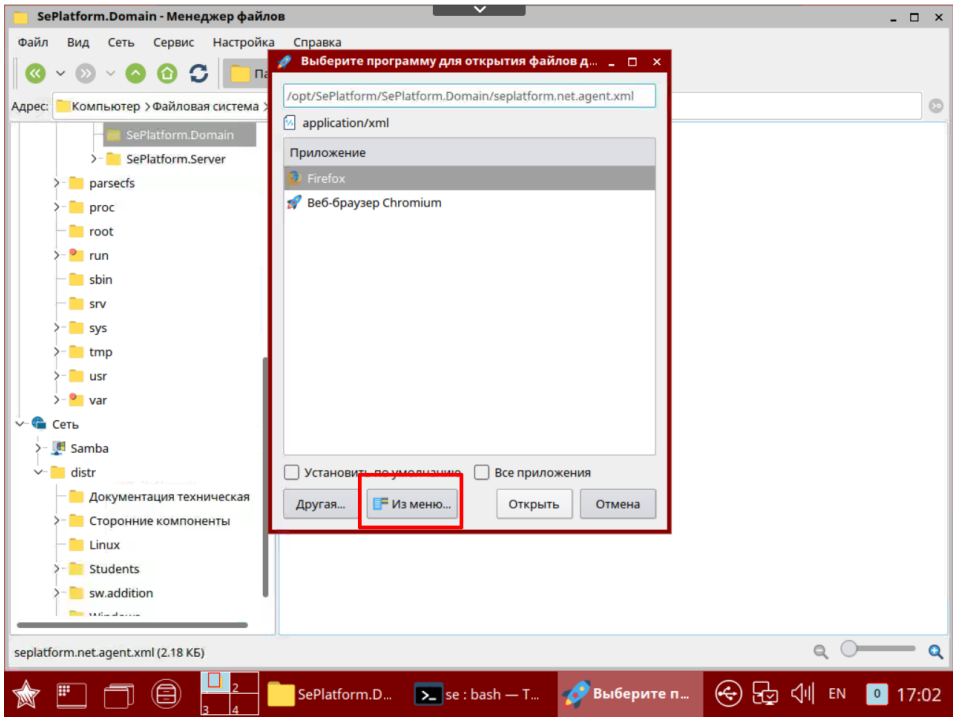
Теперь можете переходить к конфигурированию *SePlatform.Domain* на Linux машине.

5.14 Откройте папку хранения файлов конфигурации на Linux

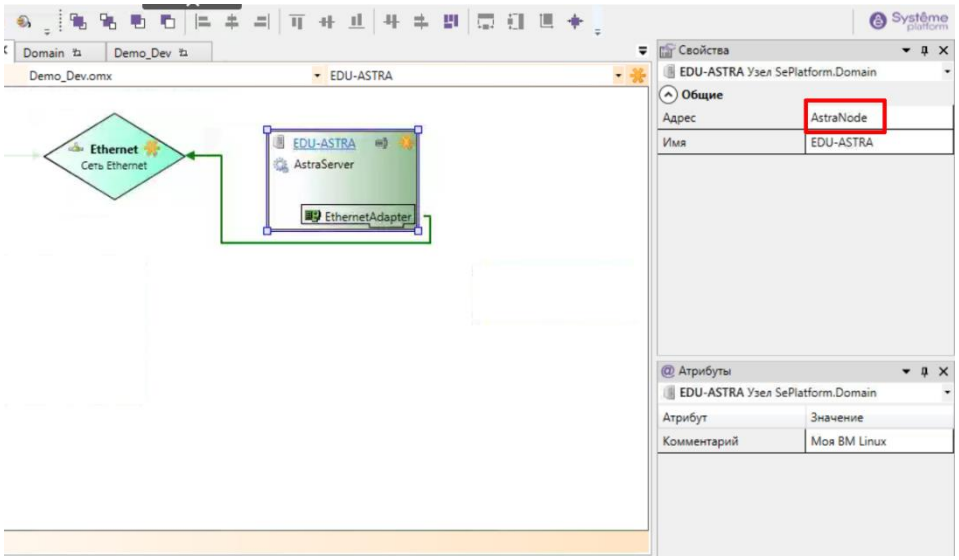
Компьютер/Файловая система/opt/SePlatform/ SePlatform.Domain



5.15 Откройте файл **SePlatform.net.agent.xml** с помощью программы «Редактор Kate». Кликните ПКМ по файлу и выберите «Открыть с помощью» далее выберите «Из меню → Офис→ Редактор Kate после чего нажимаем «Да»(для удобства можно поставить отметку «Открыть по умолчанию») в конце нажимаем «Открыть».



5.16 В атрибуте **Name** тэга **<SePlatform.Net.Agent>** введите содержимое свойства Адрес линуксового узла в среде разработки *DevStudio*.




```
<SePlatform.Net.Agent Name="AstraNode" NetEnterPort="1010" ParentAgentPort="1020">
```

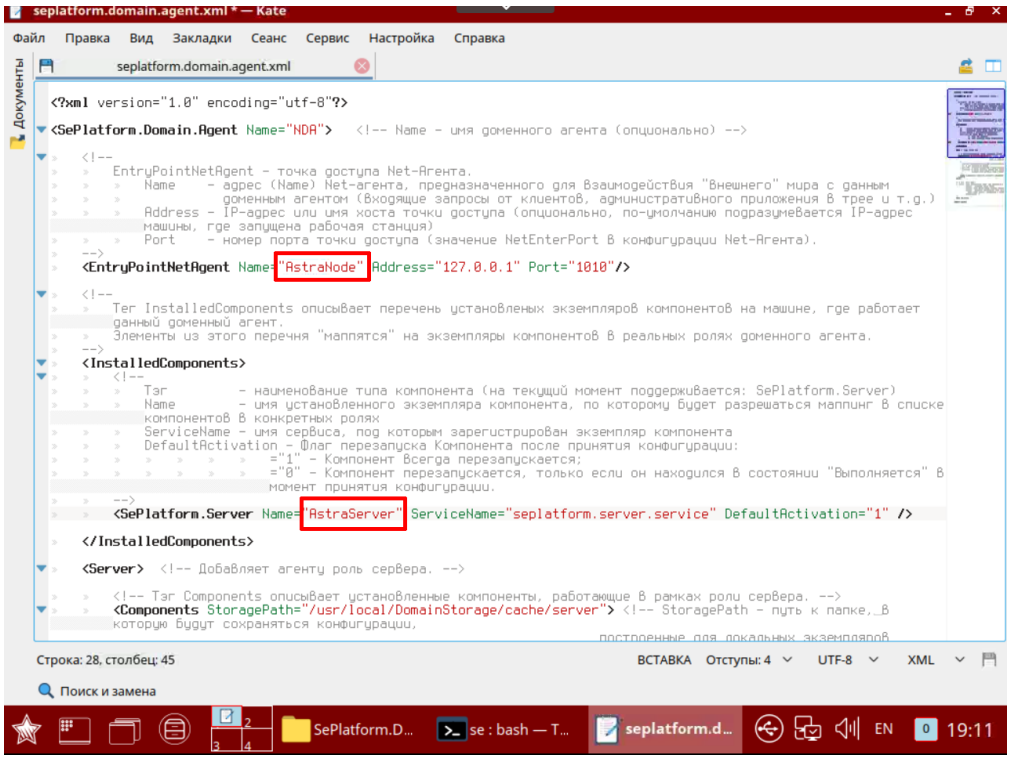
Дочерние агенты прописывать ненужно.

5.17 Сохраните файл **SePlatform.net.agent.xml** и закройте.

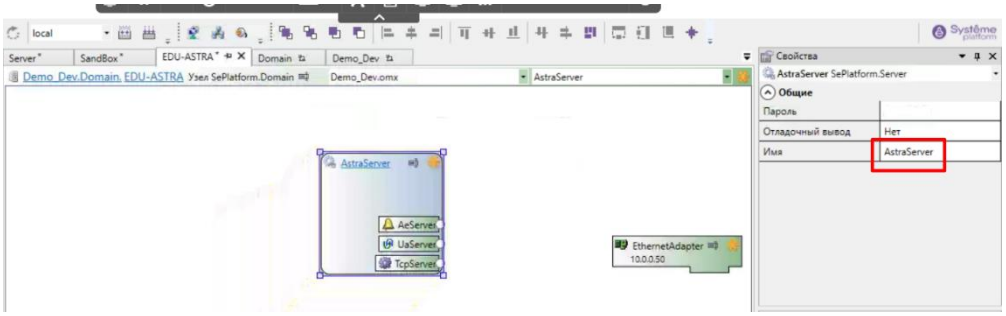
5.18 Откройте файл **SePlatform.domain.agent.xml** с помощью редактора Kate. Здесь в атрибуте **Name** тэга **EntryPointNetAgent** введите содержимое свойства Адрес линуксового узла в среде разработки *DevStudio*.

В атрибуте **ServiceName** тэга **<InstalledComponents>** описывается имя службы, которая обслуживает исполняющий компонент – элемент ***SePlatform.Server*** в проекте в *DeveloperStudio*. Эта служба в ОС Linux называется **seplatform-server.service**.

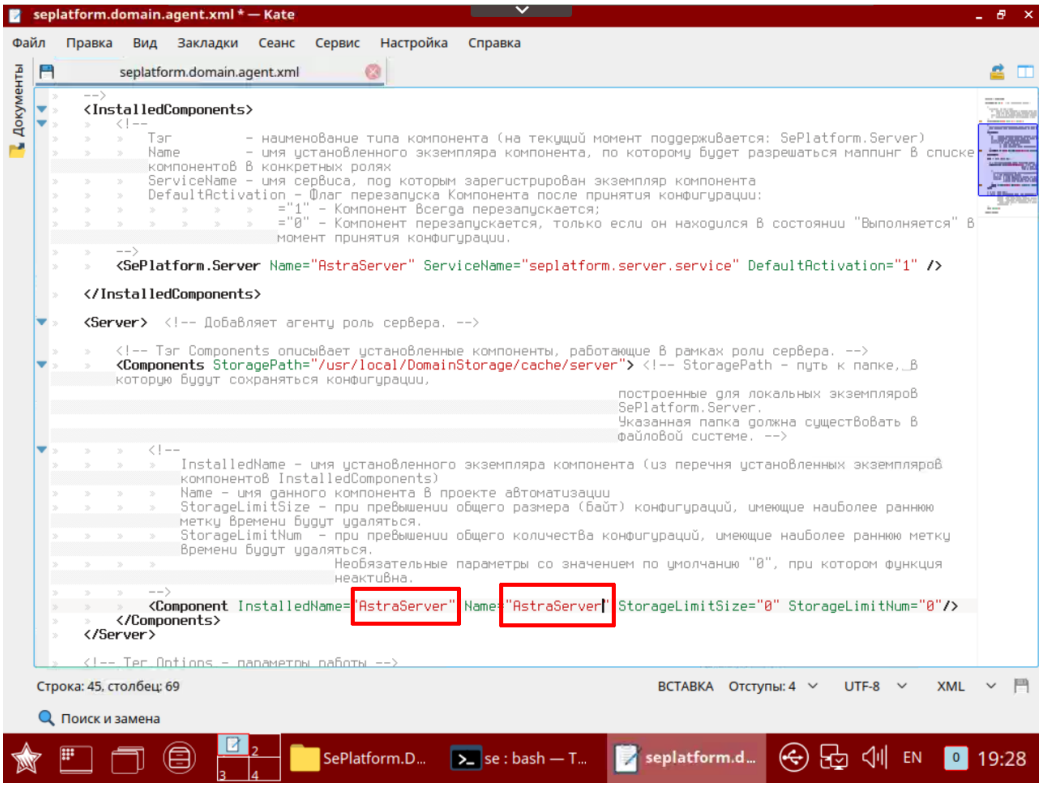
Тэг **Components** внутри **<Server>** описывает установленные компоненты, работающие в рамках роли сервера. Атрибут **Name** должен содержать имя элемента ***SePlatform.Server*** в проекте *DeveloperStudio*.



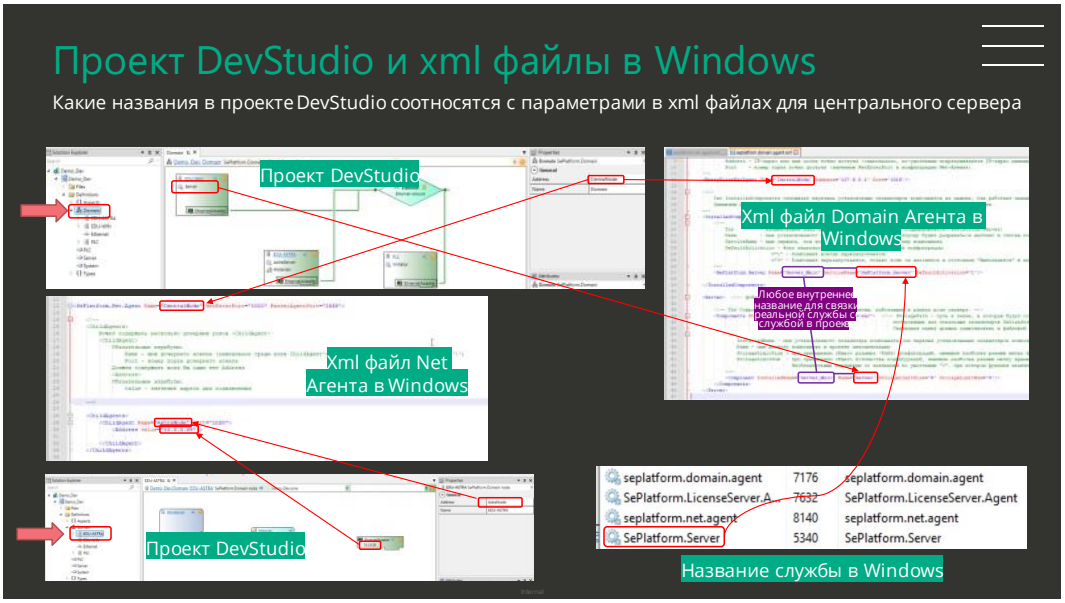
То есть служба **SePlatform-server.service** обслуживает исполняющий компонент ***AstraServer*** внутри узла, характеризующего линуксовую машину.

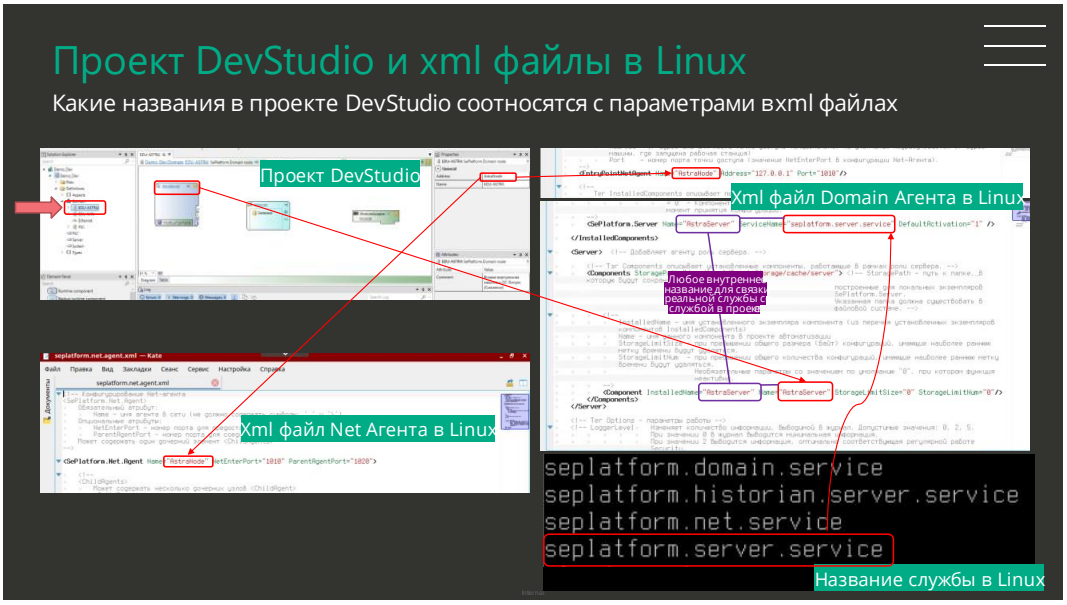


5.19 Необходимо повторить связку с сервисами. Ниже в коде в **InstalledName** и **Name** необходимо сменить имя на AstraServer



Подсказка: Ниже приведены два рисунка из краткого конспекта пользователя на которых указаны взаимосвязи проекта DevStudio с редакцией файлов xml, а также названиями сервисов в Windows и Linux:



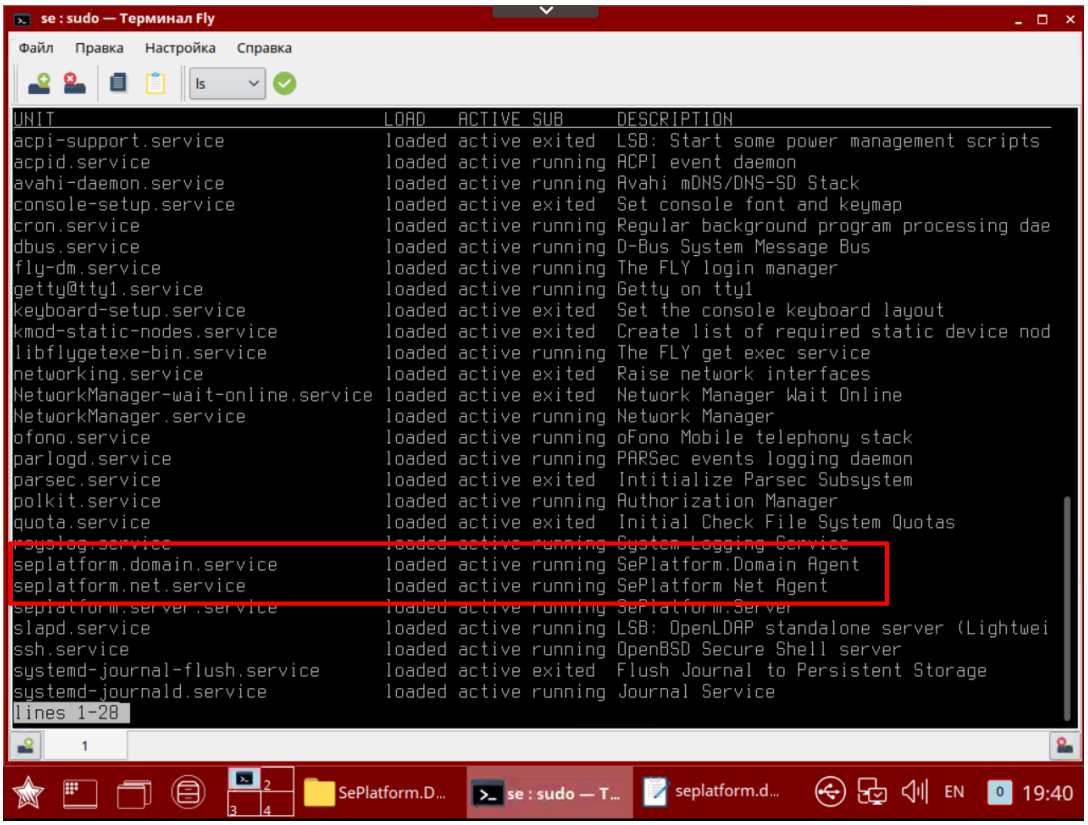


5.20 Сохраните файл **SePlatform.domain.agent.xml**. Вернитесь в командную строку

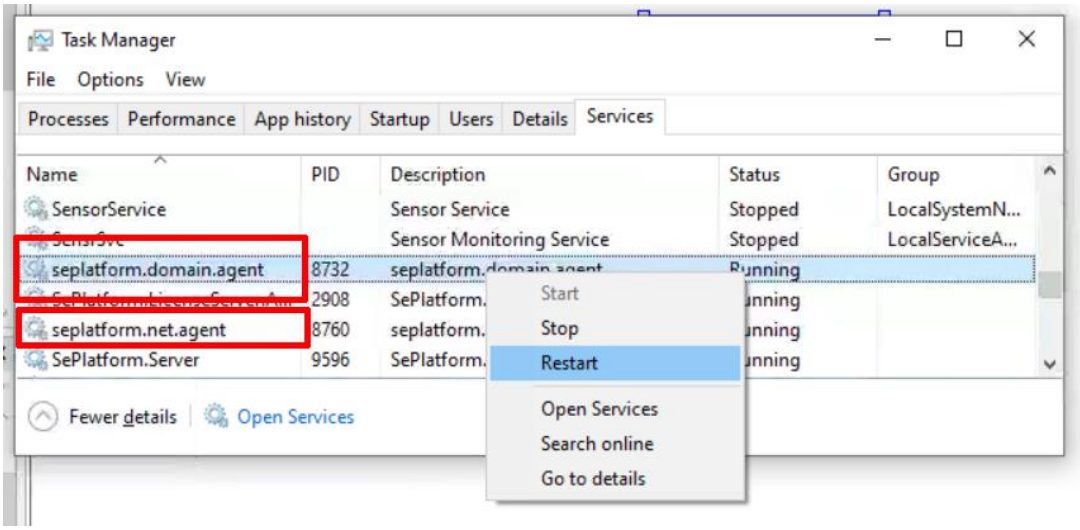
5.21 Находясь в командной строке машины с ОС Linux, введите команды для перезапуска служб **SePlatform.Net.Agent** и **SePlatform.Domain.Agent:**

sudo systemctl restart seplatform.net.service
sudo systemctl restart seplatform.domain.service

5.22 И для просмотра статуса служб **SePlatform.Net.Agent** и **SePlatform.Domain.Agent:** **sudo systemctl list-units –type service**



5.23 Те же действия нужно повторить для Windows: перезапустить службы domain и net agent



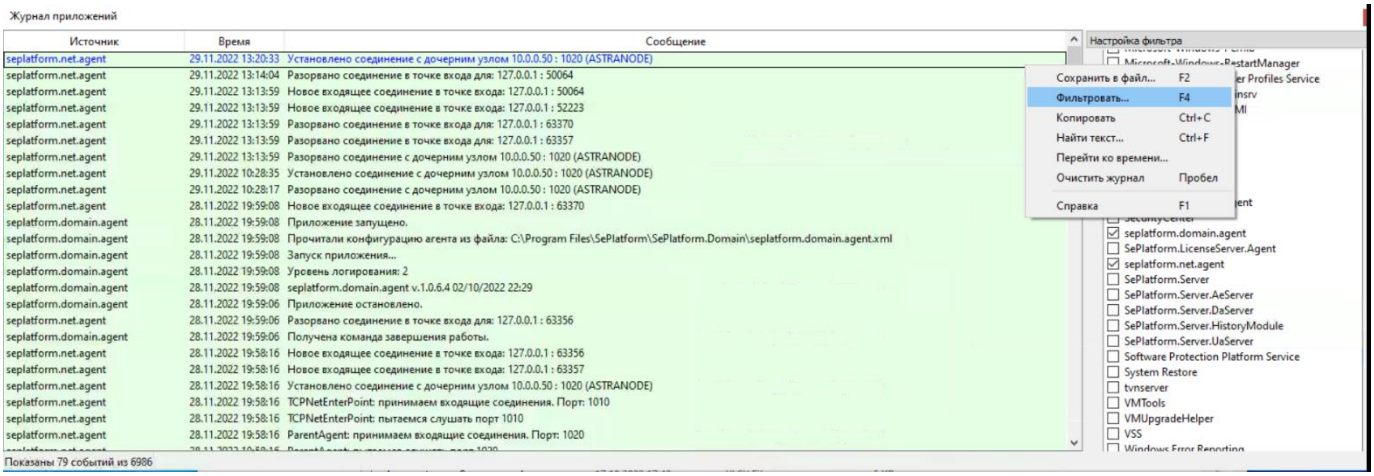
Теперь сервисы перезапущены и взаимодействуют с актуальными версиями xml файлов

6.Подключение к OPC UA Server Linux машины

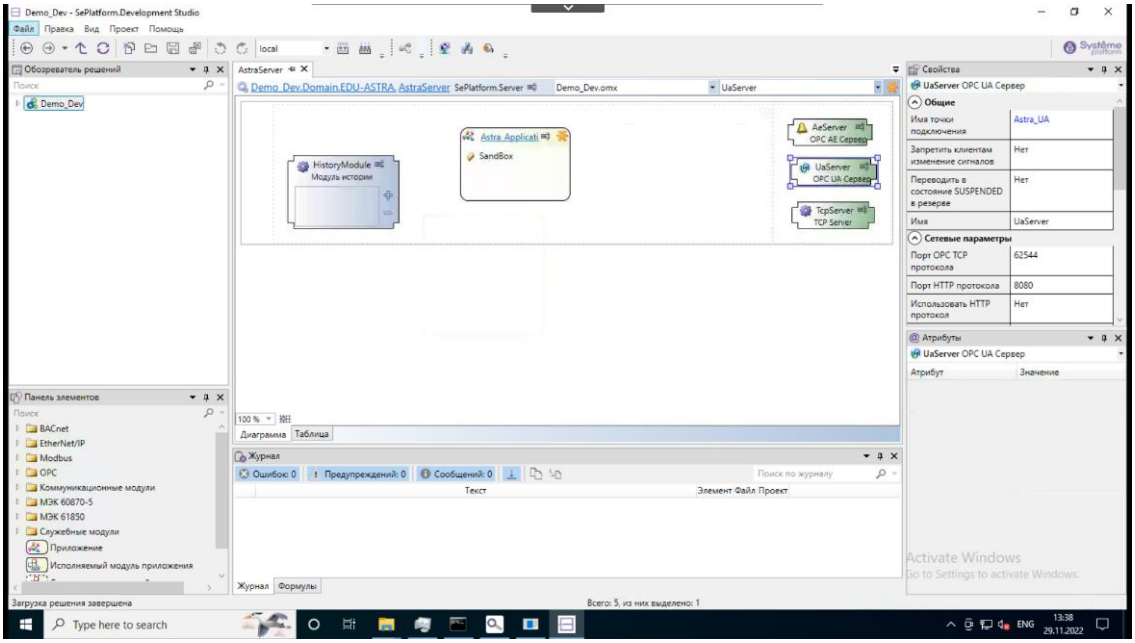
6.1 Откройте Журнал приложений(Пуск → Se Platform → Service – Log Viewer)

В нем можно увидеть сообщение, что соединение seplatform.net.agent с дочерним узлом произошло успешно, т.е связь между ВМ Windows и ВМ Linux есть

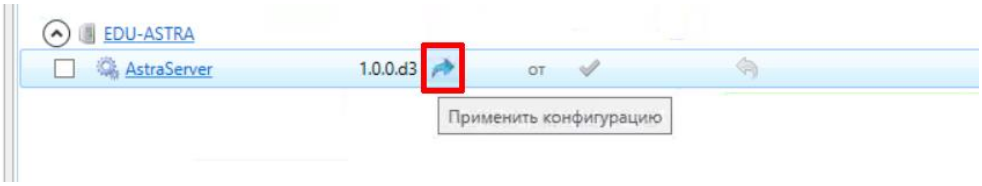
Подсказка: Для удобства просмотра сообщений можно настроить параметры фильтрации при помощи ПКМ→ Фильтровать (либо F4)



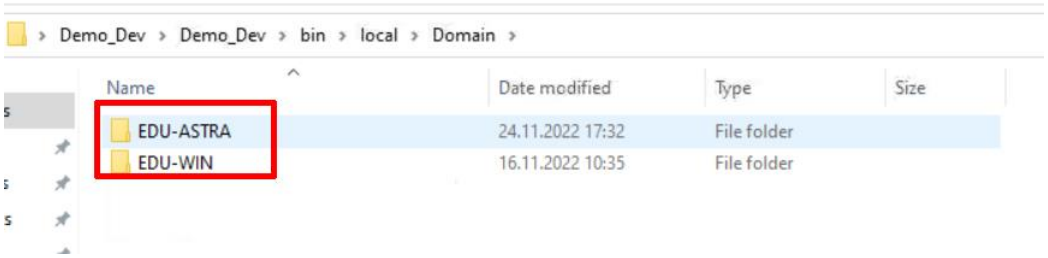
6.2 В элементе ***AstraServer*** внутри линуксового узла ***astra*** в OPC UA Сервер в свойстве «Имя точки подключения» должно бфть задано имя «Astra_UA»



6.3 Постройте решение  , перейдите к развёртыванию  и примените конфигурацию к линуксовому серверу.

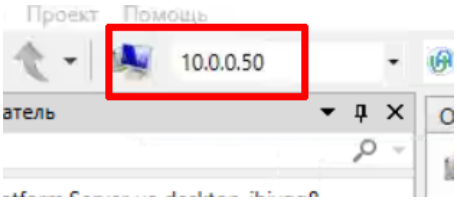


6.4 Теперь в папке конфигурации на рабочем столе появились две подпапки с файлами конфигурации для Windows и Linux:

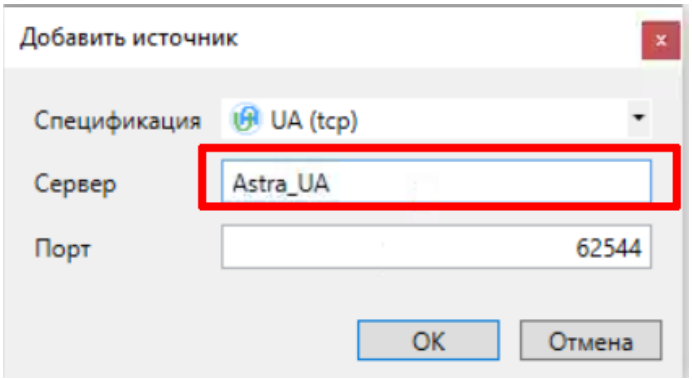


теперь есть возможность загружать конфигурации для обоих серверов

6.5 Откройте OpcExplorer, введите IP-адрес Linux машины, подтвердите ввод нажатием клавиши Enter и нажмите на иконку монитора слева – Запрос проверки связи.



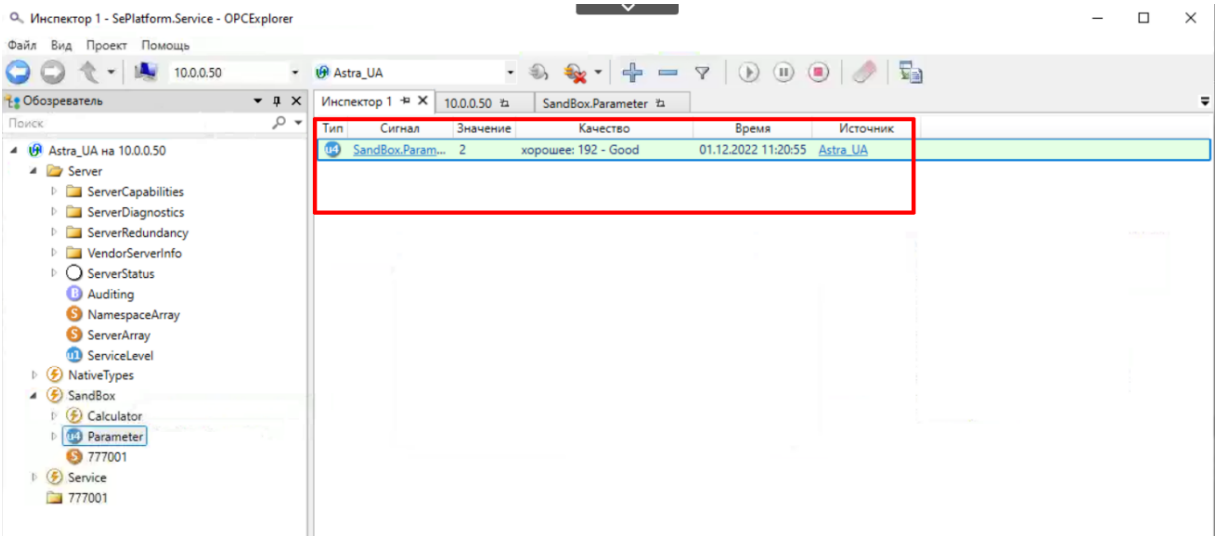
6.6 Нажмите на кнопку «Добавить источник»  и введите данные для подключения как на изображении ниже (в соответствии с проектом DevStudio), нажмите ОК.



6.7 В поле «Источник данных» выберите нужный источник и подключитесь к нему (кнопка «Подключиться к исполняющему компоненту» ).

6.8 Добавьте в окно Инспектор ***Parameter*** из объекта ***Sandbox***. При изменении значения сигнала, меняется и качество. Подробная информация описана в документации (Документация → SePlatform.Server → SePlatform.Server → Сигналы SePlatform.Server).

Мы можем увидеть, что источником нового сигнала является Linux сервер

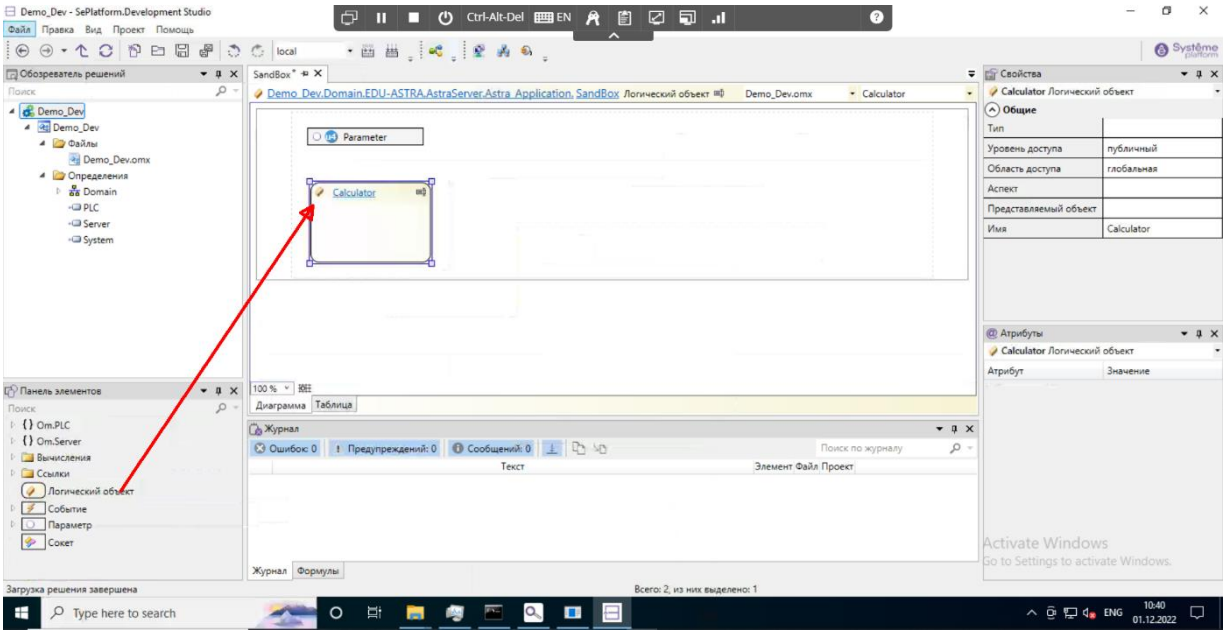


7. Модификация проекта SePlatform.DevStudio

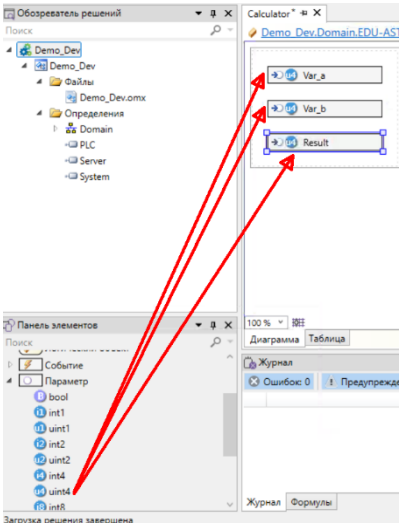
Работа с атрибутами

Познакомимся атрибутами объектов и параметров.

7.1 Перейдите в ***SandBox*** при помощи Обозревателя решений DevStudio. Перетяните сюда ***Логический объект*** из Панели элементов. Назовите его Calculator.

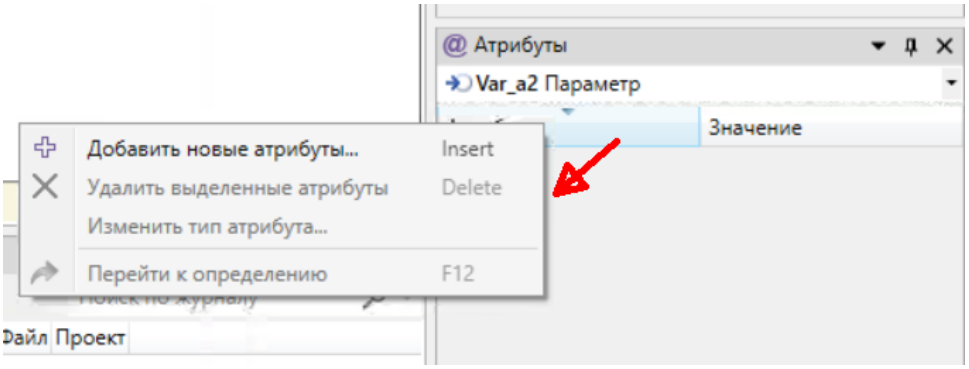


7.2 Перейдите внутрь и добавьте сюда ***3 параметра*** типа Uint4: Var_a, Var_b, Result.

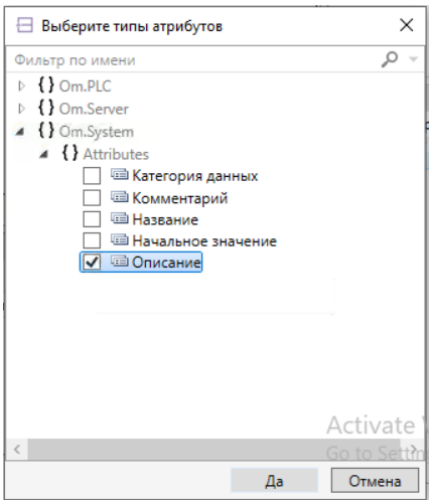


7.3 Добавьте атрибут Описание и Начальное значение для ***Var_A***:

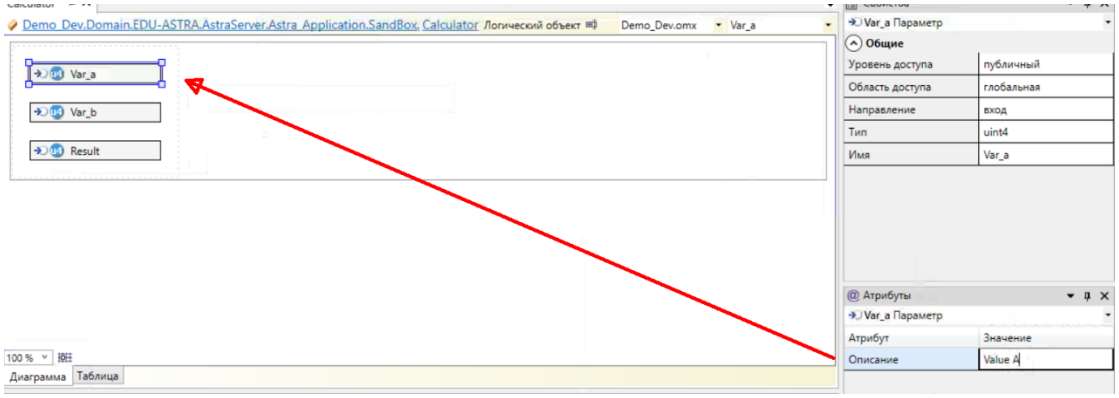
- выделите ***Var_A***, кликните ПКМ по Панели атрибутов → Добавить новые атрибуты



- выберите атрибут Om.System → Attributes → Описание



- В описании пропишите «Value A».

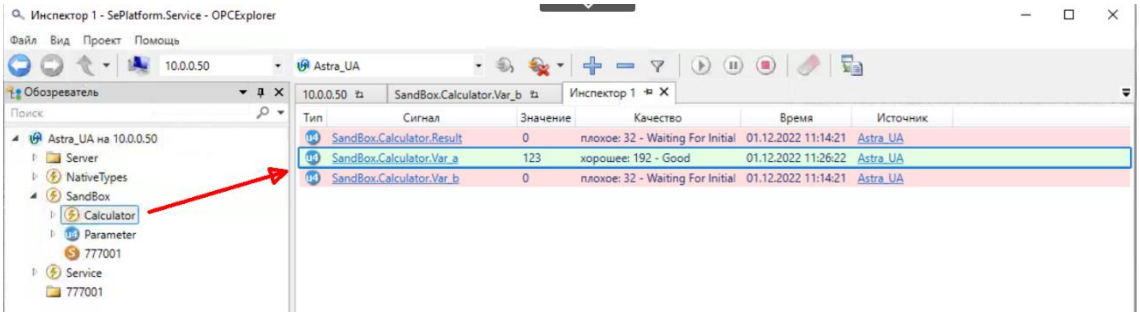


7.4 Добавьте для ***Var_b*** и ***Result*** атрибут описания: Value B и Result.

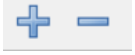
Подсказка: можно сразу добавить атрибуты сразу к нескольким параметрам. Для этого надо выделить их, ПКМ → добавить атрибуты. Так же можно написать общую часть названия

7.5 Постройте решение, перейдите к развёртыванию и примените конфигурацию к линуксовому серверу.

7.6 Перейдите в ***OpcExplorer***, добавьте в Инспектор весь объект ***Calculator***. Здесь отобразилось начальное значение у параметра ***Var_A*** и атрибуты описания для каждого из параметров.



Таким образом в ***DeveloperStudio*** можно добавлять атрибуты не только параметрам, но и объектам.

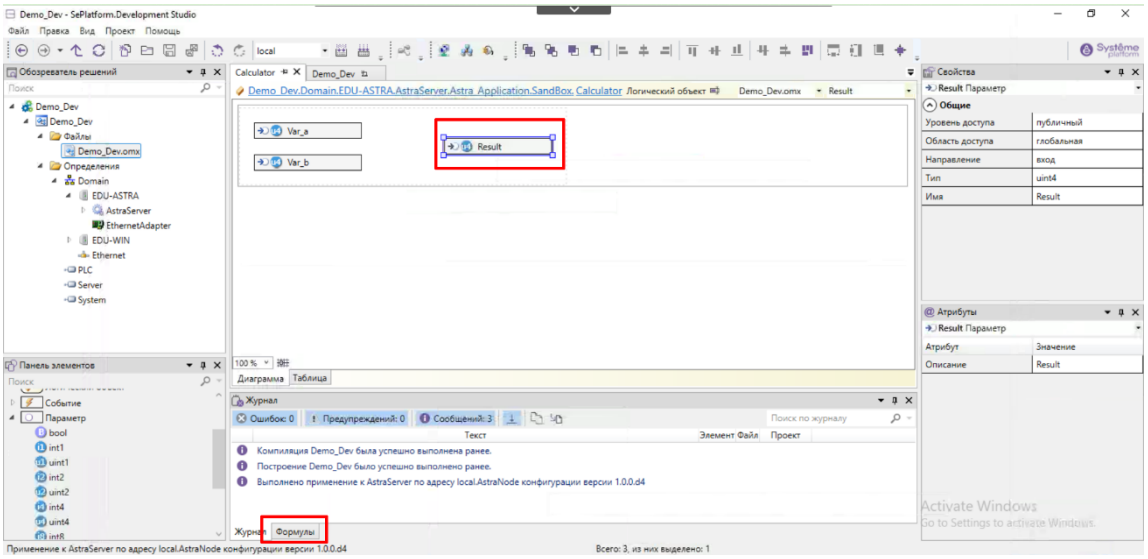
Подсказка: можно очистить список нажав на иконку «ластик»  , чтобы управлять отображаемыми параметрами можно их выделить, а затем добавить/убрать при помощи иконок 

Работа с логикой

Теперь попробуем построить небольшую логику в проекте, научим объект ***Calculator*** вычислять сумму двух переменных разными способами. Вычисления сервер производит с помощью модуля логики. В данном модуле используется скриптовый язык SePlatform.OM. Язык SePlatform.Om позволяет описывать исполняемые элементы скриптов (процедуры и формулы), которые запускаются в процессе работы компонентов *SePlatformPlatform*. Скрипты можно создавать в процессе работы со следующими программными продуктами: *SePlatform.Server*, *SePlatform.HMI*, *SePlatform.DevStudio*. Подробное описание можно найти в документации в разделе **SePlatform.Om**.

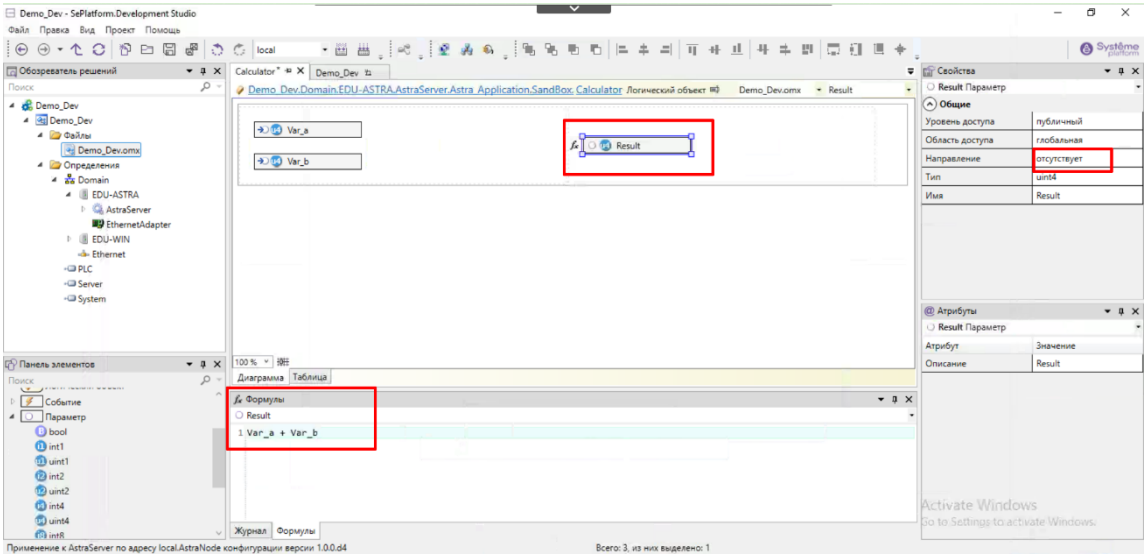
В сервере выполнить операции можно различными способами. Один из способов – это заставить параметры самостоятельно вычислять свои значения.

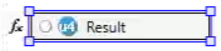
7.7 Выберите ***параметр*** Result и перейдите во вкладку Формулы (здесь указываются однострочные скрипты, не требующие точки с запятой на конце).



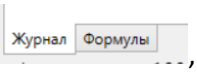
7.8 Во вкладке Формулы введите скрипт, помещающий в переменную ***Result*** сумму переменных: **Var_a + Var_b** (пользуйтесь подстановкой значений из выпадающего списка).

Подсказка: если в поле формулах нет возможности ввода, то необходимо сменить направление переменной на «Выход» либо «Отсутствует»



Около переменной ***Result*** появился знак функции 

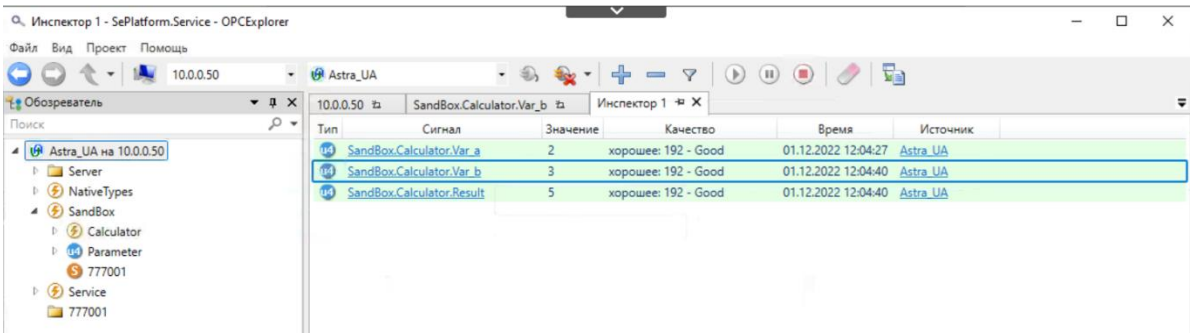
, который говорит о том, что данная переменная содержит в себе какую-то формулу. На нее можно навести курсор мыши, чтобы узнать подробности о функции.

Подсказка: После введения формулы, переключайтесь на журнал 

, чтобы у вас была возможность видеть информацию об ошибках

7.9 **Постройте решение**, **перейдите к развёртыванию** и **примените конфигурацию** к линуксовому серверу.

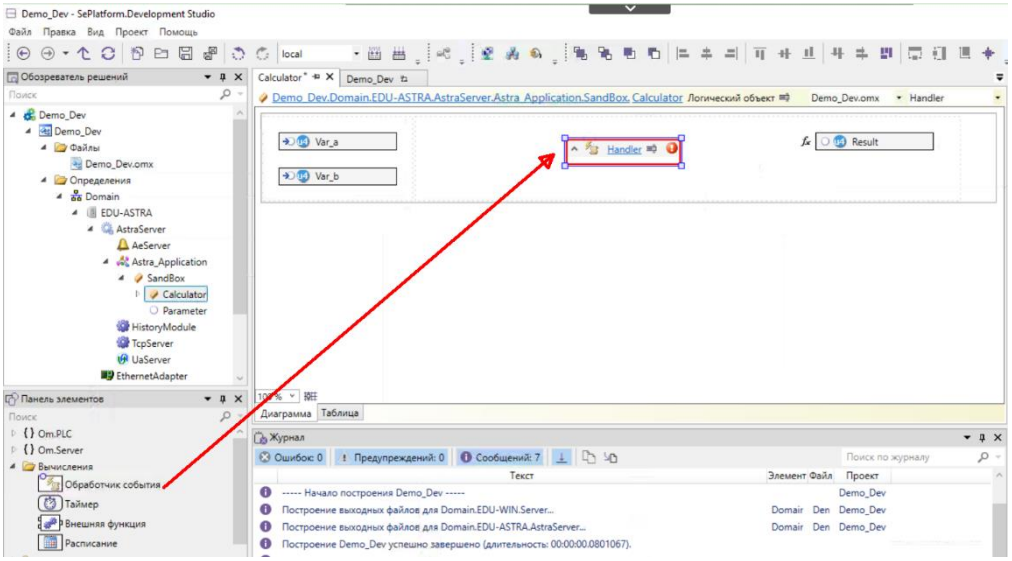
7.10 Перейдите в ***OpcExplorer***, дождитесь переподключения Astra_UA. Проверьте работу калькулятора: введите значения для ***переменных A и B***.



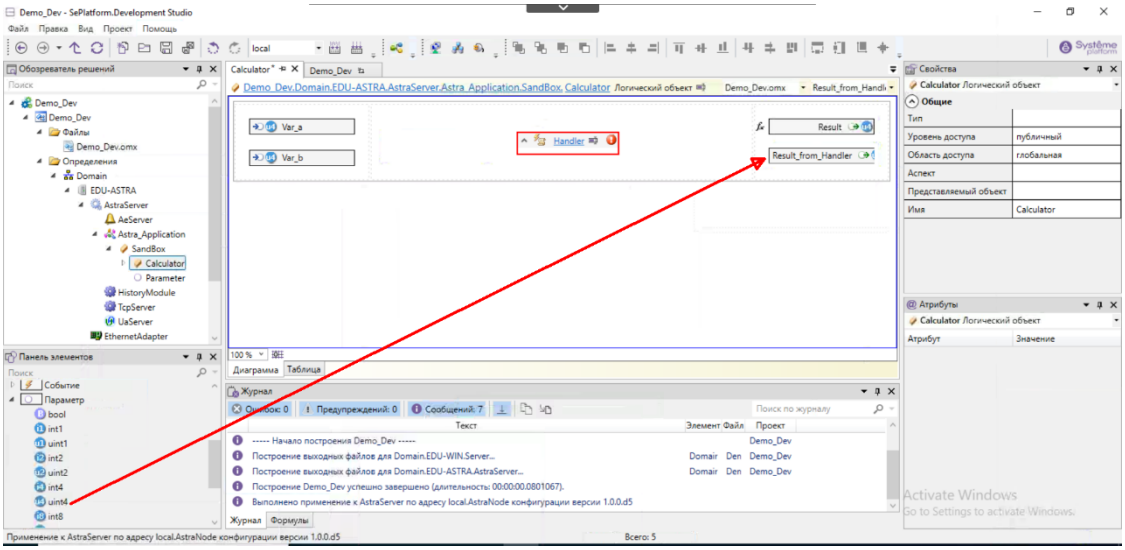
Важно! Начальные значения, которые указаны в атрибутах параметров не являются триггерами для запуска вычислений. Вычисления запускаются во время выполнения.

Следующий способ выполнения операций в сервере – производить вычисления при помощи обработчика событий.

7.11 В панели элементов разверните вкладку Вычисления и перетяните на рабочее поле элемент ***Обработчик события***, (Handler)

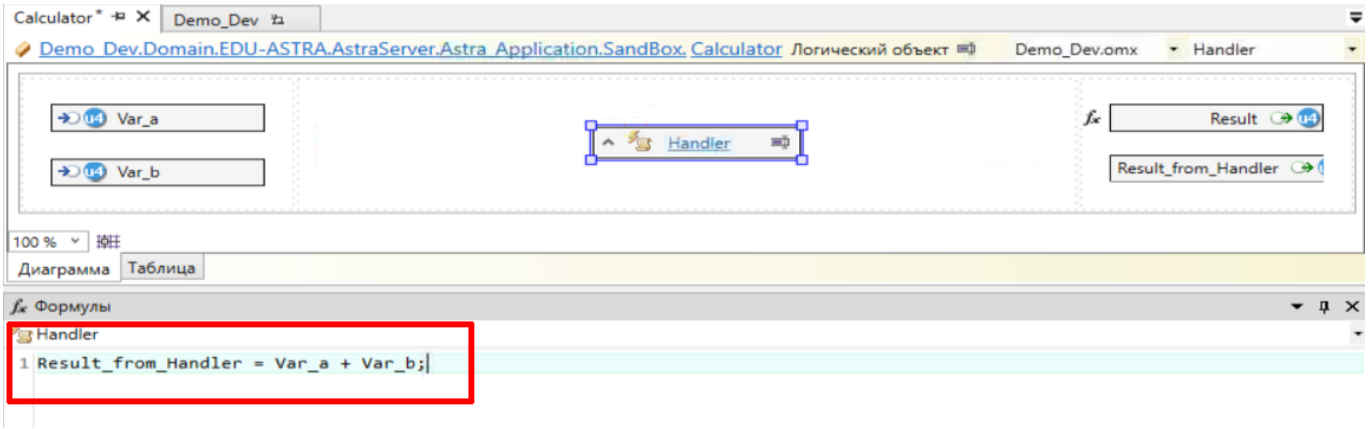


7.12 Добавьте в объект ***Calculator*** ещё один ***параметр*** типа Uint4, назовите его Result_from_Handler (результат из обработчика).

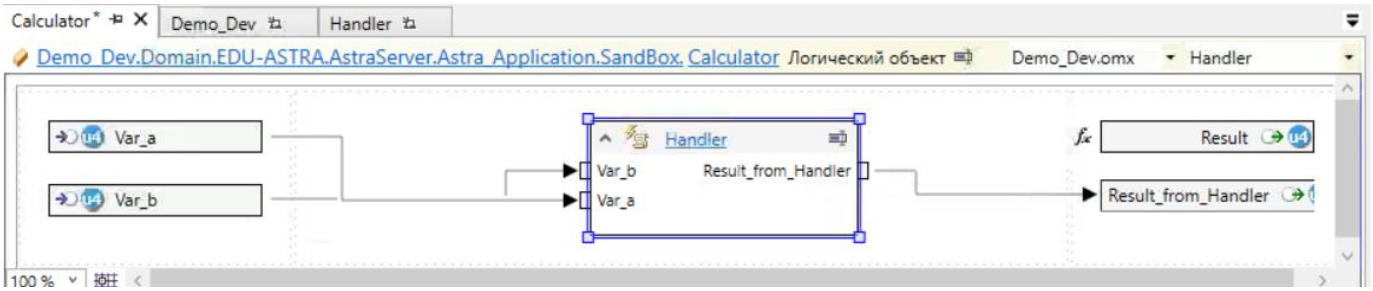


7.13 Выделите ***Handler***, перейдите во вкладку Формулы и запишите скрипт, который в переменную ***Result_from_Handler*** помещает сумму ***переменных a и b*** *(Как на рисунке ниже)*. Обратите внимание, что в конце необходимо указать «;».

Подсказка: если дважды кликнуть по Handler появится диалоговое окно, в котором также можно писать скрипты.

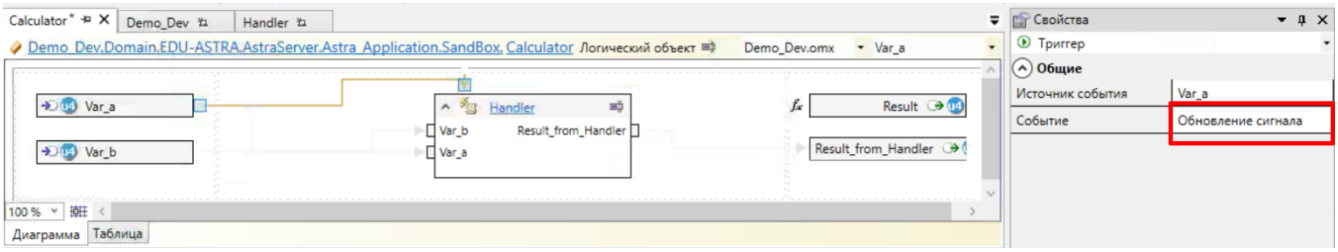


7.14 Вернитесь к журналу, обновите страницу (F5) для отрисовки связей.



7.15 Сам по себе обработчик не работает, его нужно активировать. Запускаться он будет при изменении параметров ***Var_a*** и ***Var_b***.

Чтобы реализовать его работу наведите мышкой на ***переменную a***, дождитесь возникновения окружностей справа и слева прямоугольника, наведите мышкой на правую окружность – курсор превратится в прицел, захватите эту окружность и тяните связь к левому верхнему углу ***обработчика*** (на нём тоже появится окружность).



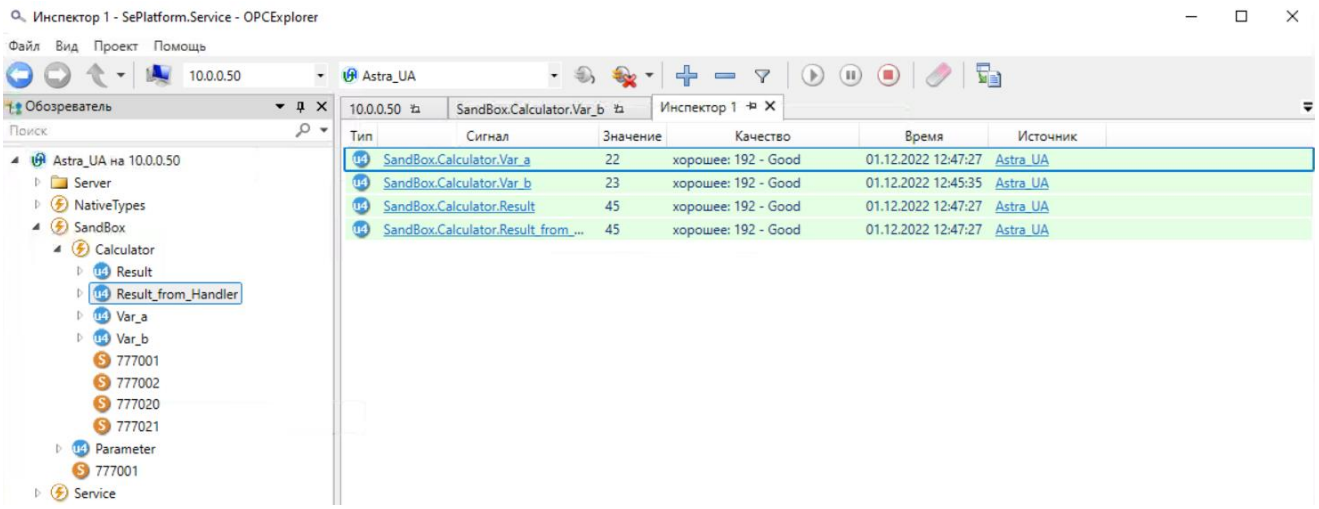
Данные связи называются ***триггерами*** обработчика. При нажатии на оранжевую стрелку появится панель свойств данного триггера и в свойстве «Событие» можно выбрать, при каком действии, произошедшем с сигналом, будет срабатывать ***Обработчик события***.

Важно! Нельзя «триггерить» обработчик от его выходных параметров, сервер такое вычисление не

произведёт.

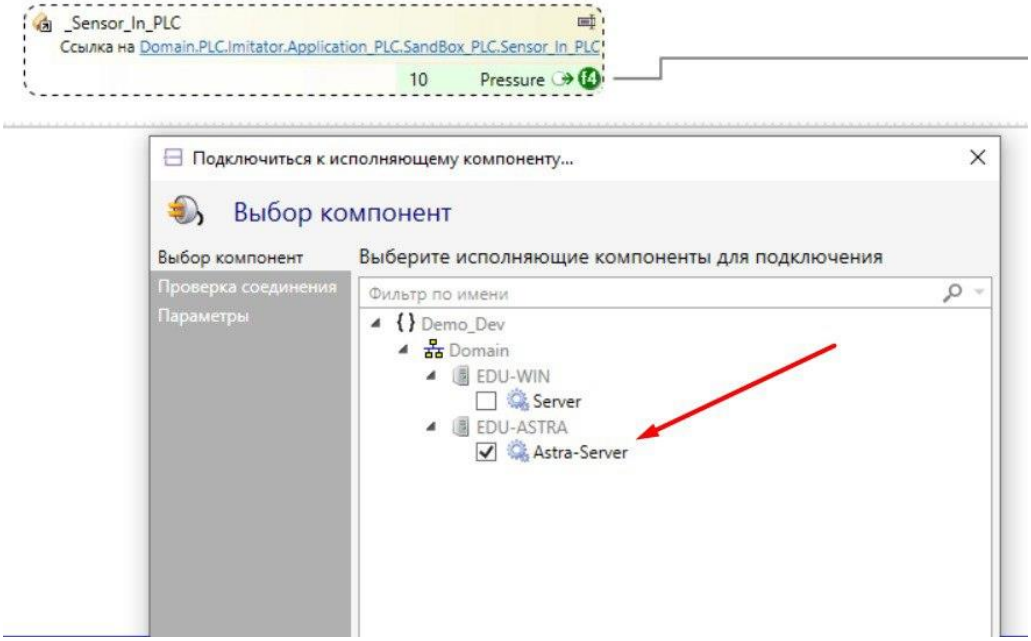
7.16 Постройте решение, перейдите к развёртыванию и примените конфигурацию к линуксовому серверу.

7.17 Перейдите в *OpcExplorer*, дождитесь переподключения Astra_UA и добавьте в инспектор недостающий сигнал ***Result_from_Handler***. Проверьте работу калькулятора: введите значения для переменных a и b.



Обратите внимание, что если ввести данные в Переменную b, значение ***Result_from_Handler*** не изменится, т.к. триггером является изменение значения переменной a. Если ввести значение для Var_a, то ***Result_from_Handler*** посчитает сумму.

Подсказка: после повторной загрузки проекта в OPC Explorer необходимо отключаться и подключаться вновь, чтобы отобразились актуальные данные



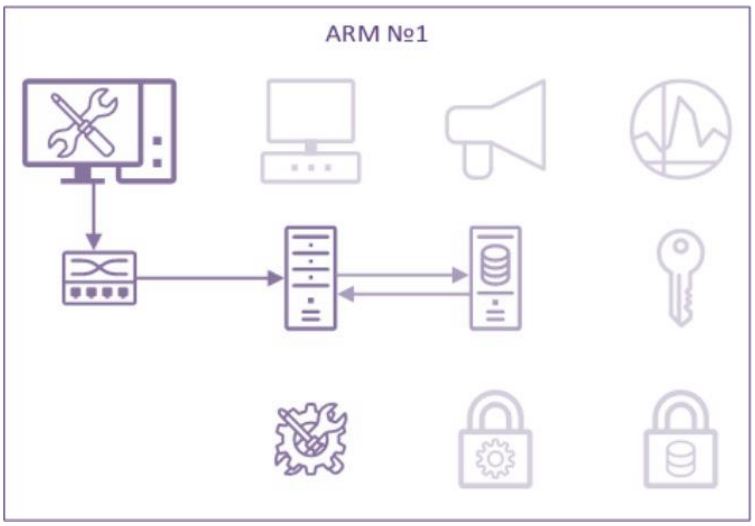
8. Работа с компонент Systeme Platform платформы – SePlatform.Historian

SePlatform.Historian – компонент Systeme Platform платформы для сбора и сохранения информации о технологическом процессе.

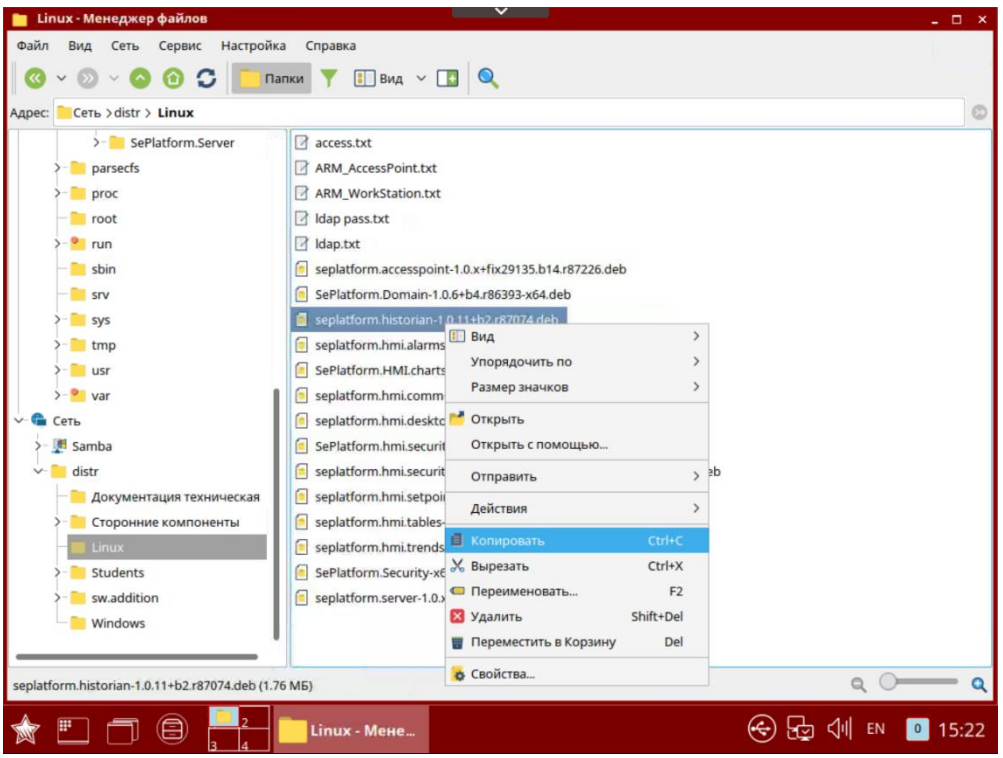
Возможности *SePlatform.Historian*:

- ✓ Сбор и хранение оперативных значений параметров технологического процесса;
- ✓ Сбор и хранение истории событий и тревог технологического процесса;
- ✓ Предоставление исторических данных клиентам.

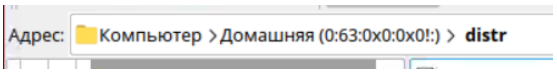
Перед началом работы, необходимо установить компонент *SePlatform.Historian* на машину с ОС Linux.



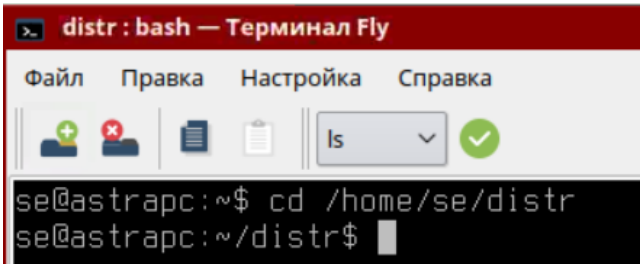
8.1 Перейдите в ВМ Linux, скопируйте дистрибутив из сетевой папки на вашу ВМ



8.2 Вставьте файл в папку с дистрибутивами

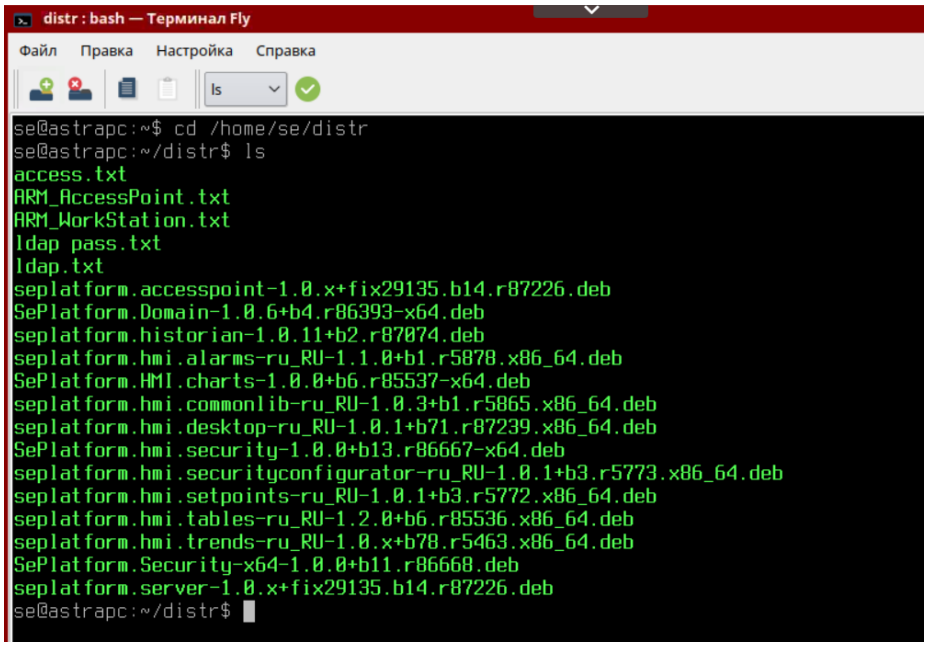


8.3 Откройте терминал Fly и переместитесь в папку distr

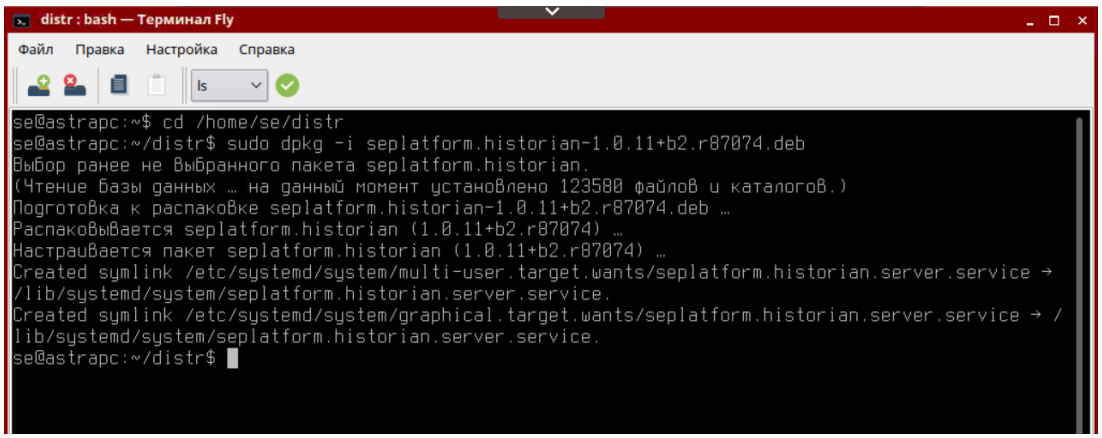


8.4 Удостоверьтесь, что Вы находитесь в директории

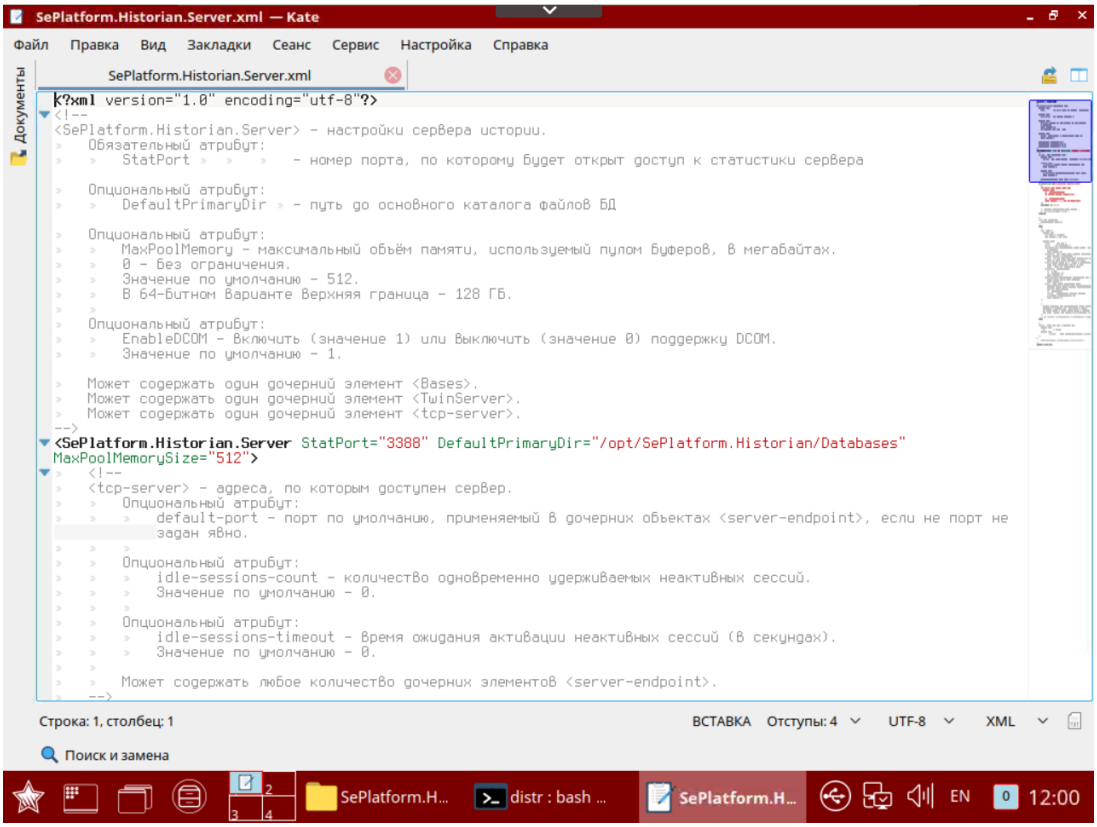
Подсказка: С помощью команды ls в Linux можно посмотреть какие файлы находятся в папке



8.5 Для установки модуля **SePlatform.Historian** введите команду: **sudo dpkg –i SePlatform.historian*******.deb**.



8.6 Для конфигурирования *SePlatform.Historian* перейдите в папку /opt/SePlatform/SePlatform.Historian и откройте файл **SePlatform.Historian.Server.xml** при помощи Редактора Kate.



На машинах с ОС Windows *SePlatform.Historian* может работать как через DCOM, так и по TCP. Так как данный проекты кроссплатформенный, нужно отключать DCOM.

8.7 Для отключения DCOM в строку с тэгом **SePlatform.Historian.Server** необходимо дописать EnableDCOM="0", тем самым отключив поддержку DCOM.

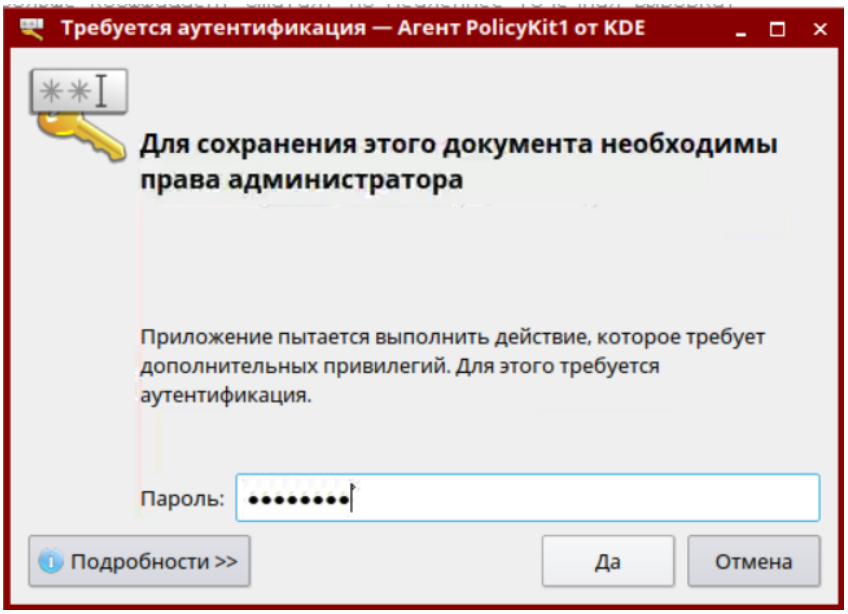
```
Может содержать один дочерний элемент <Bases>.
Может содержать один дочерний элемент <TwinServer>.
Может содержать один дочерний элемент <tcp-server>.
-->
<SePlatform.Historian.Server StatPort="3388" DefaultPrimaryDir="/opt/SePlatform.Historian/Databases"
MaxPoolMemorySize="512" EnableDCOM="0">
<!--
<tcp-server> - адреса, по которым доступен сервер.
Опциональный атрибут:
```

8.8 Спуститесь ниже к тэгу **<Base>**. Перед закрытием данного тэга необходимо раскомментировать строчку для определения базы данных. Удалите комментарий и в атрибуте **Alias** введите название для данной базы – «History».

```
-->
<Base Alias="History" PreferredCommonCacheLimit="128" ActiveStorageDepth="3" StorageDepth="365" />
</Bases>

<!--
```

8.9 Сохраните файл **SePlatform.Historian.Server.xml** и закройте.

Система может запросить права администратора. Для этого укажите пароль вашей ВМ Linux – **12345678**

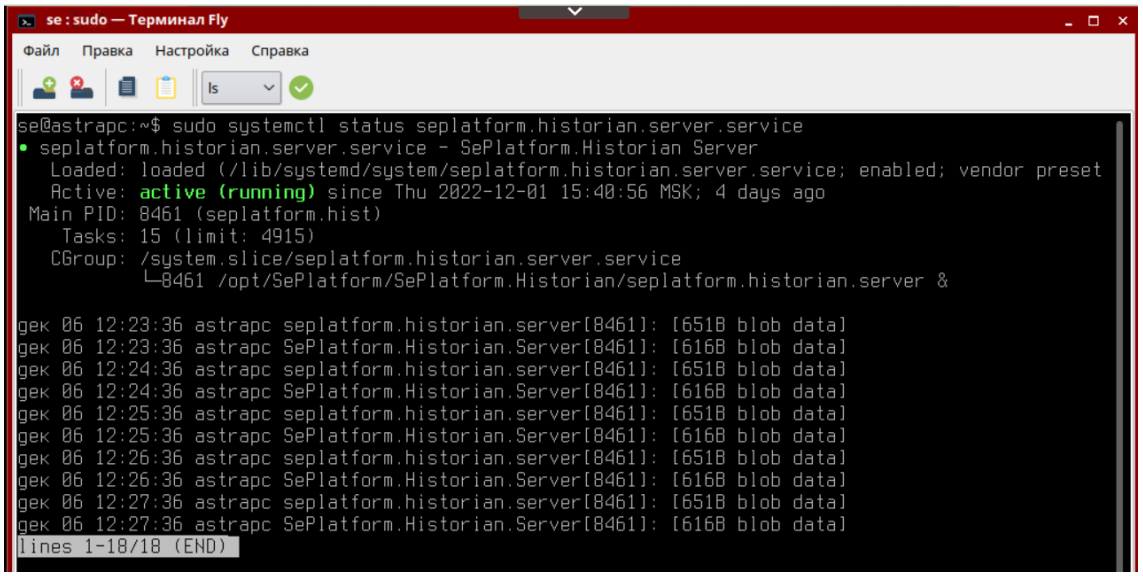


8.10 Выйдите в Терминал и введите команды для перезапуска службы

sudo systemctl restart seplatform-historian.server.service

и отображения статуса.

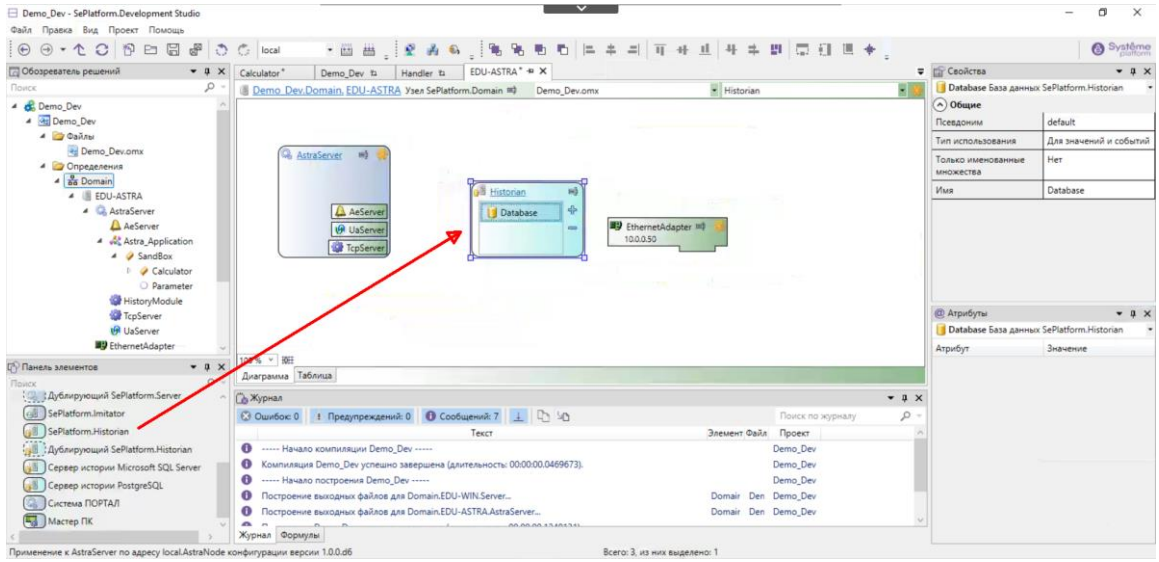
sudo systemctl status seplatform-historian.server.service



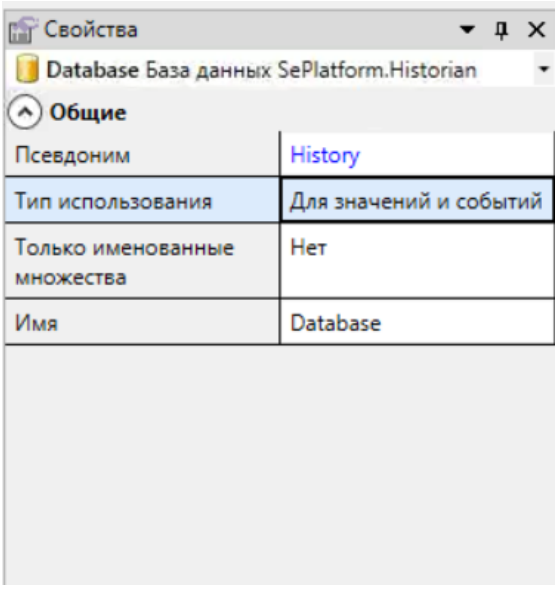
За сохранение истории отвечает атрибут истории. Добавим нескольким параметрам возможность сохранять историю и просмотрим оперативные и исторические графики с помощью *SePlatform.Trends*. Вернитесь в Windows → *DevStudio*, и перейдите в машину Linux. В панели элементов выберите и перетащите на рабочую область блок SePlatform.Historian

Теперь необходимо настроить проект автоматизации для того, чтобы описать, где находится *SePlatform.Historian*.

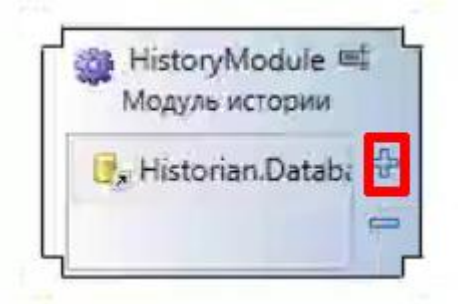
8.11

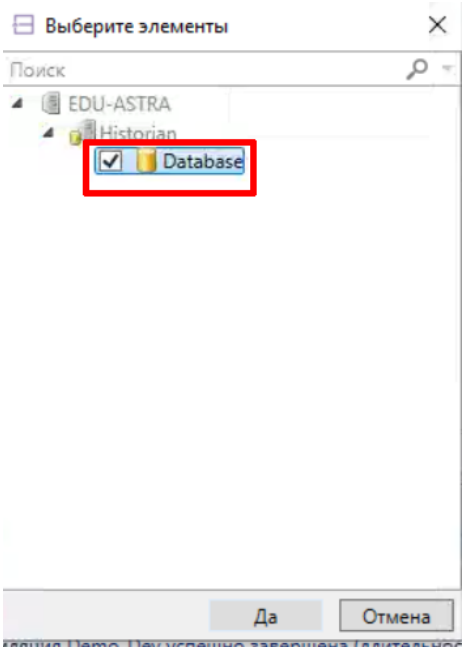


8.12 В поле «Псевдоним» укажите «History»

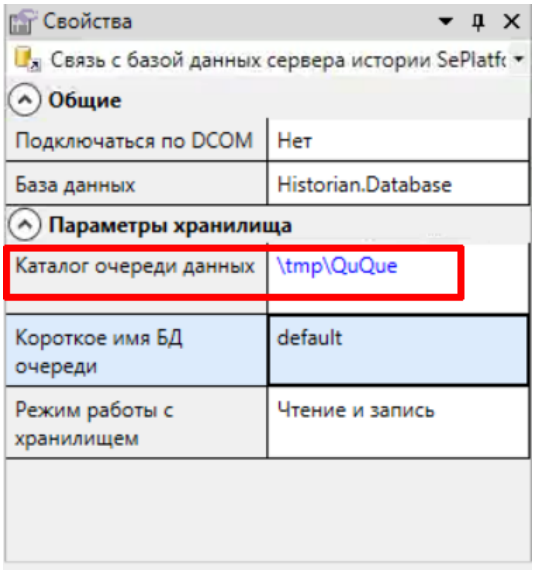


8.13 Укажите базу данных, с которой будет работать модуль истории. Перейдите Astra Server → в окне модуля нажмите «+» и выберите из выпадающего списка базу данных, с которой будет работать модуль. После чего нажмите «Да»





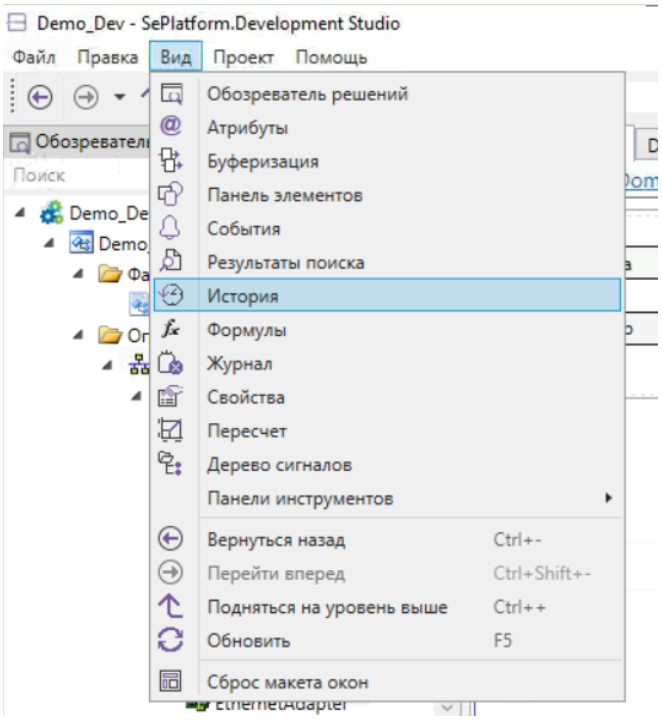
8.14 Нажмите мышью на базу данных и в окне свойств укажите Каталог очереди данных: \tmp\QuQue



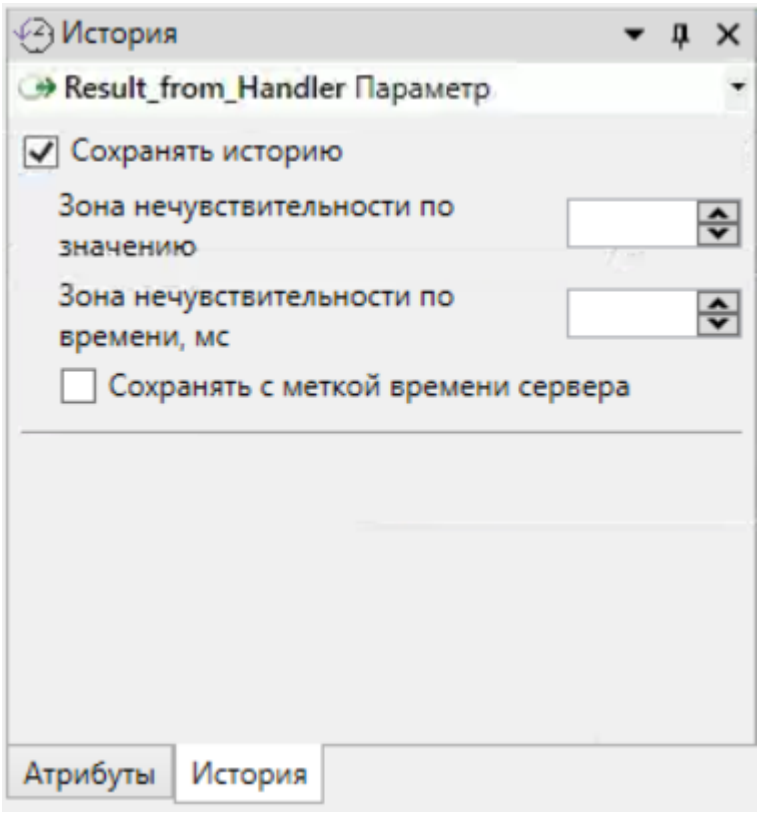
Важно! Данный каталог выбирается только для нужд обучения. При разработке реального проекта необходимо выбрать другой каталог.

8.15 Перейдите Sandbox → Calculator и настройте сохранение данных для переменной Result_from_Handler.

Для этого в меню файл выберите → Вид → История

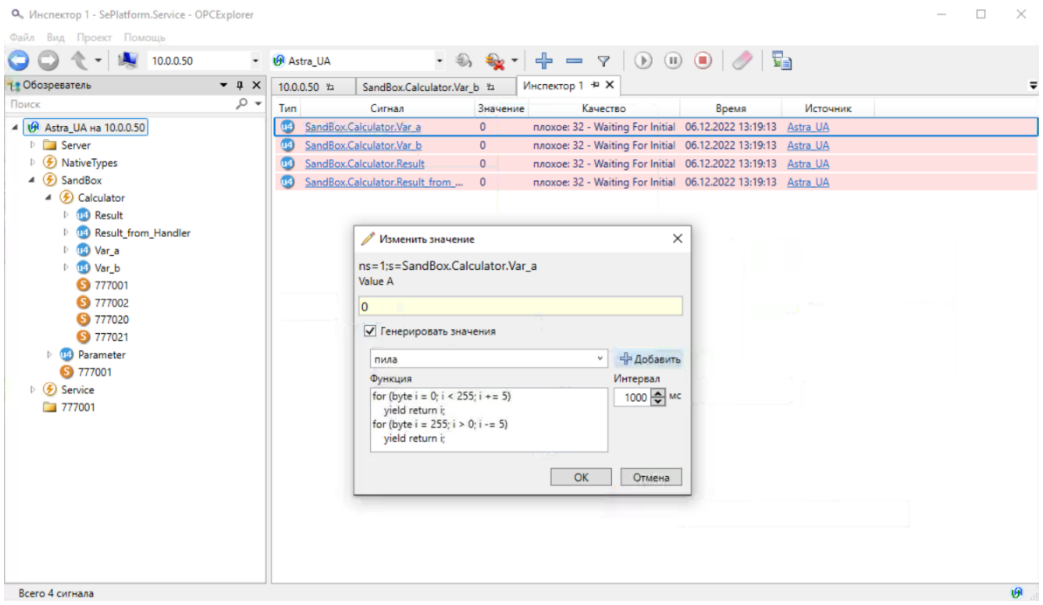


8.16 В появившемся окне поставьте галочку «Сохранять в историю»

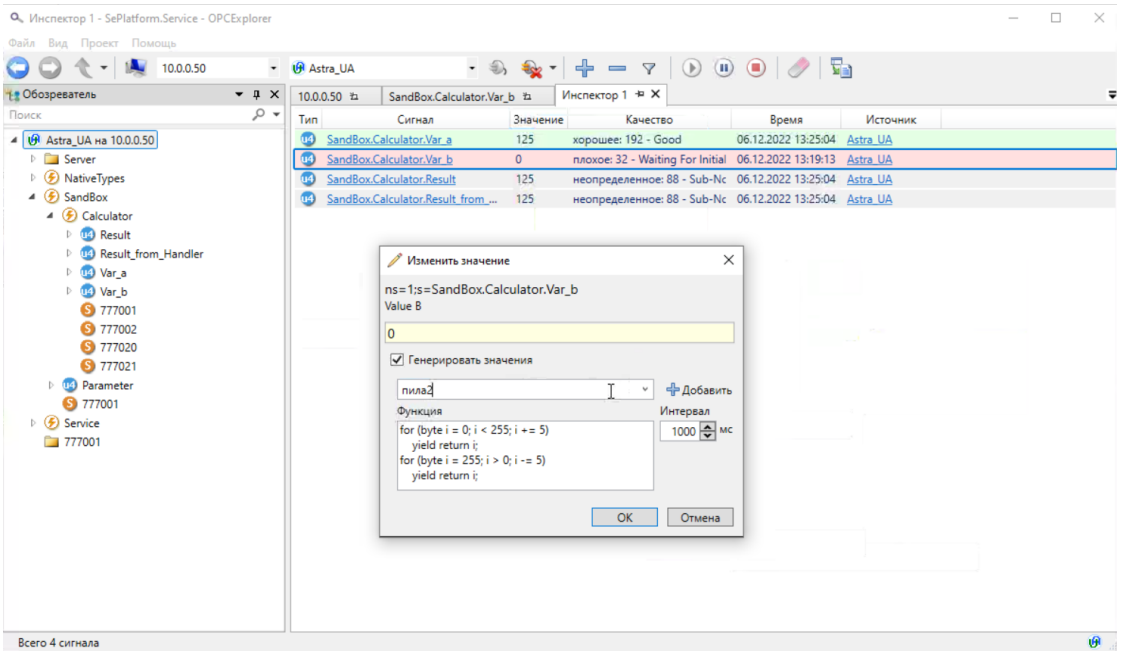


8.17 Постройте решение и залейте конфигурацию

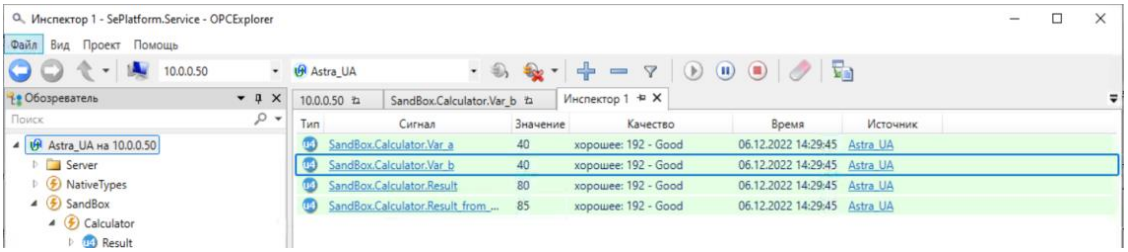
8.18 Перейдите в OPC Explorer, кликните дважды ЛКМ по параметру значение и отметите галочкой «Генерировать значения» Выберите из выпадающего списка ниже генератор «пила».



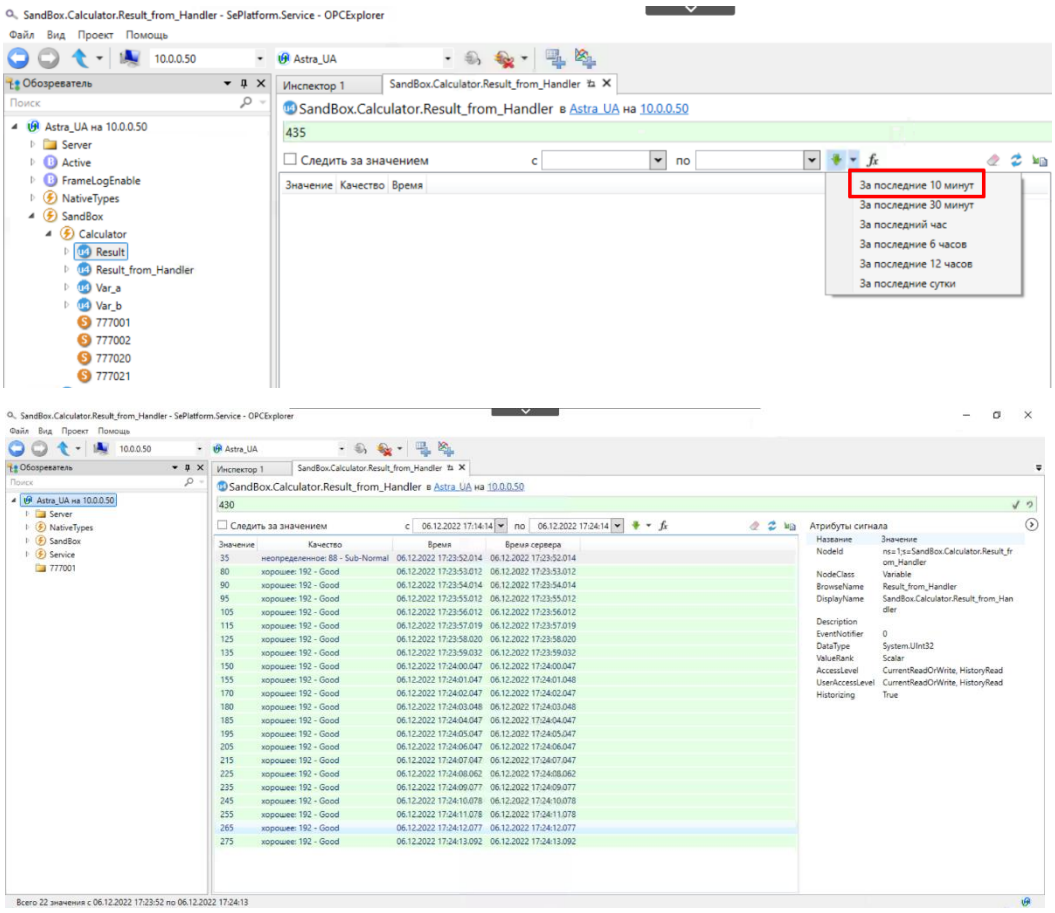
8.19 Проделайте то же самое для переменной Var_b и укажите другое имя «Пила2»



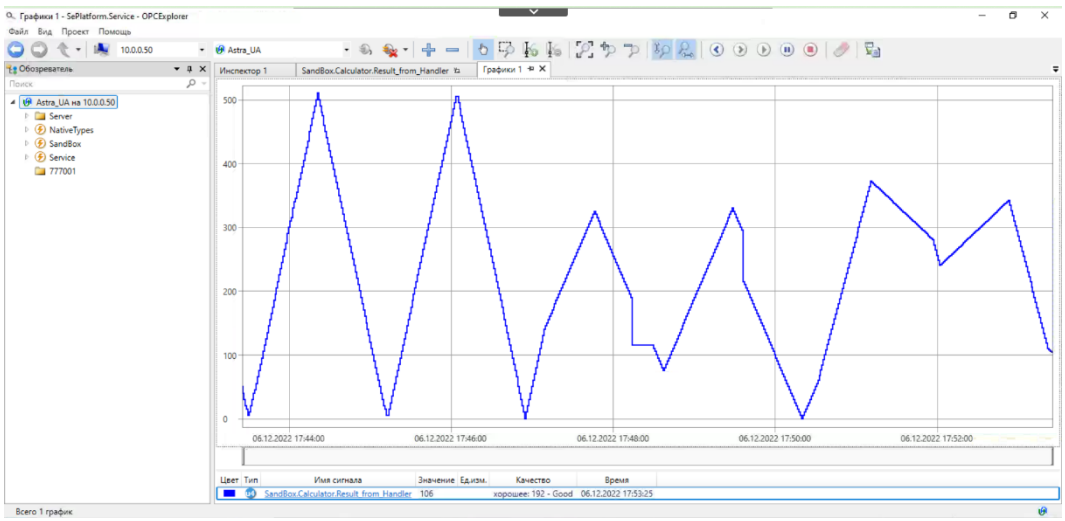
8.20 Теперь значения генерируются, а история сохраняется в Result_from_Handler



8.21 Для просмотра исторических данных кликните ЛВК по параметру ***Result_from_Handler***. В открывшемся окне есть возможность указать интервал времени, за которое необходимо просмотреть историю – кнопка **Загрузить историю значений за выбранный интервал времени**. Выберите «За последние 10 минут».



Также можно открыть график сигнала. Для этого в Обозревателе кликните ПКМ по нужному параметру и выберите. Показать график сигнала.

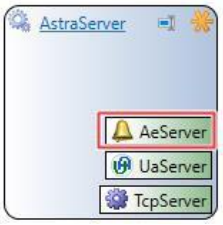


9. Работа с событиями

В проекте есть возможность настроить генерацию событий и тревог при наступлении определённых условий. Например, при превышении уровня можно уведомлять об этом пользователя не только цветовой индикацией на мнемосхеме, но и с помощью событий в журнале. В том числе, со звуковым сопровождением. Журналы событий затем можно анализировать при разборе инцидентов. Сначала, необходимо настроить генерацию событий в сервере.

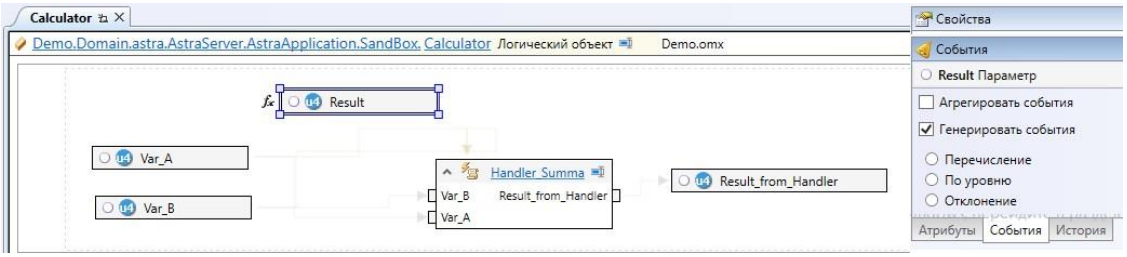
Настройка генерации событий в SePlatform.DevStudio

Для генерации событий необходимо, чтобы в составе сервера был модуль **OPC AE Server**, который будет генерировать события и передавать их клиентским приложениям. В конфигурации по умолчанию уже присутствует данный модуль.

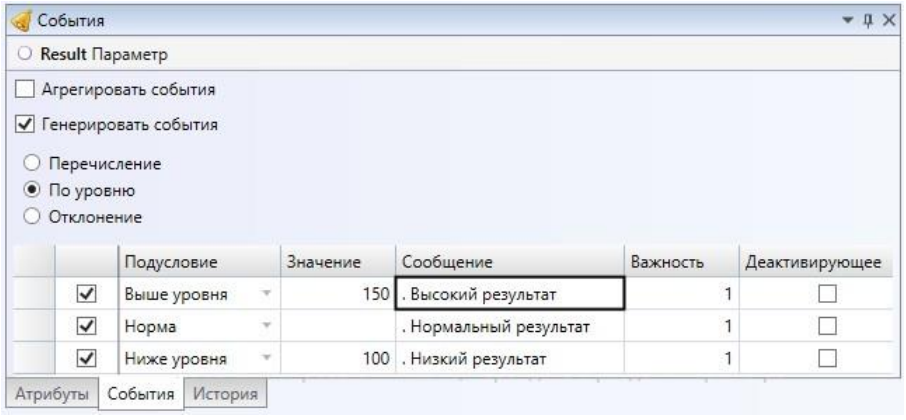


Теперь необходимо сконфигурировать сигналы, которые будут генерировать события. Для примера будем использовать подусловие отклонении значения сигнала суммы двух сигналов.

9.1 Перейдите в *DevStudio*, нажмите Вид → События для отображения соответствующей вкладки на панели атрибутов (в правой нижней части экрана). Перейдите в логический объект ***Calculator***, выделите параметр ***Result***, откройте вкладку События и поставьте галочку около Генерировать события.



9.2 Выберите событие По уровню, укажите значение и сообщение, которое будет отображаться при возникновении данного события.



9.3 Поднимитесь на уровень выше (в логический объект SandBox), задайте атрибут Описание для логического объекта ***Calculator*** – Калькулятор

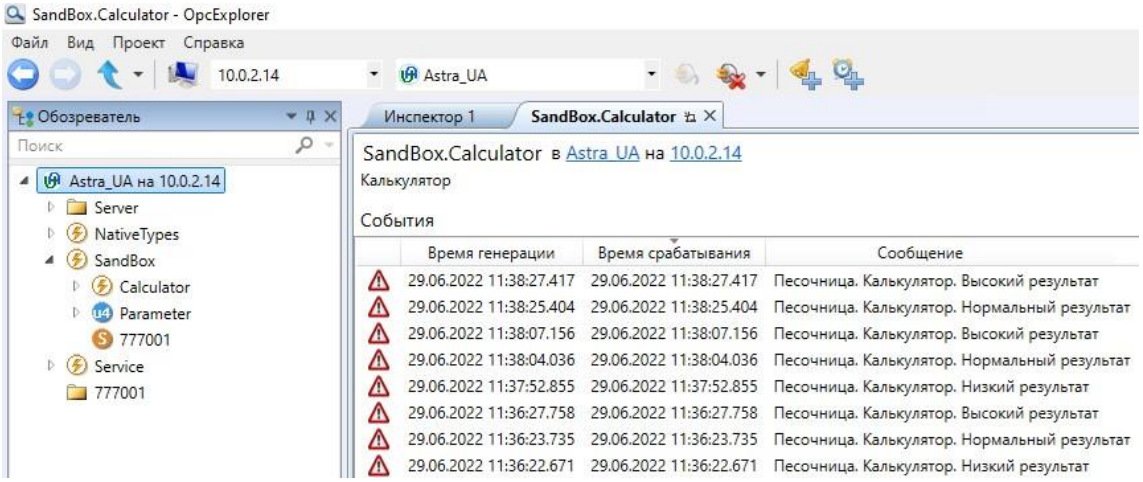
9.4 Поднимитесь ещё на уровень выше (внутрь AstraApplication), задайте атрибут Описание для логического объекта ***SandBox*** – Песочница.

9.5 **Постройте решение**, **перейдите к развёртыванию** и **примените конфигурацию** к линуксовому серверу.

Просмотр событий

В *OpcExplorer* есть несколько способов наблюдать события.

1. Перейдите в *OpcExplorer*, дождитесь переподключения сервера.
2. Так как события настроены внутри логического объекта ***Calculator***, нажмите на него в Обозревателе. События по уровню генерируются тогда, когда порог пересекается, а не каждый раз, когда значение выше порога.



Есть ещё один способ просмотра событий в *OpcExplorer*. Кликните на вкладку Проект → Добавить события. Перетяните в открывшееся окно объект ***Calculator*** из обозревателя.

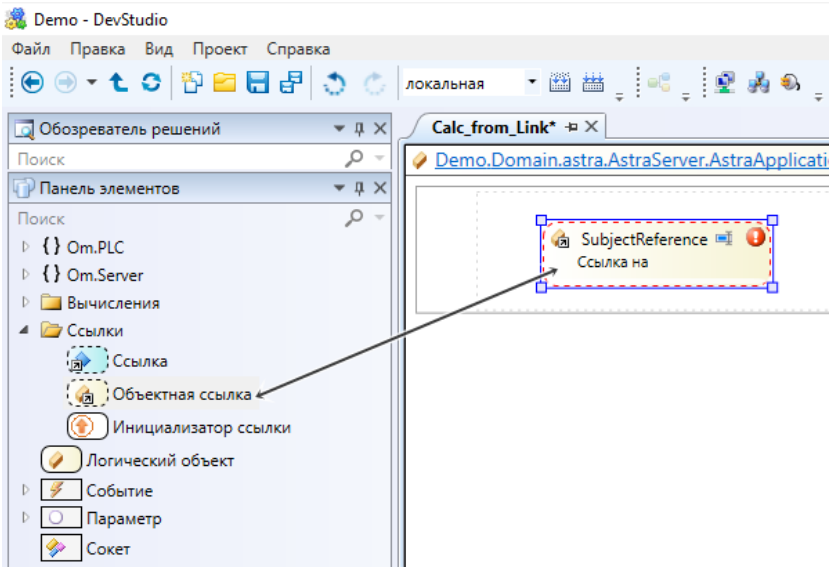
8. Модификация проекта SePlatform.DevStudio (часть 2)

Далее углубимся в возможности *SePlatform.DevStudio*. Начнём с обмена данными между объектами. Изучим функционал объектных ссылок. Для корректной работы демонстрационного проекта отключите генераторы с ***Var_A*** и ***Var_B*** *OpcExplorer*.

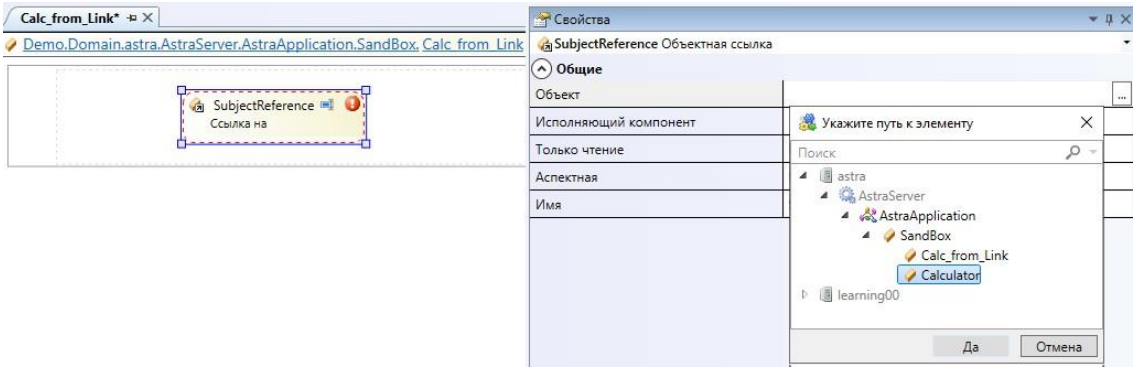
Передача данных между объектами (ссылки)

Для начала в сервер нужно добавить ещё объект, с которым будет происходить обмен данными.

1. Перейдите в ***SandBox*** и из панели элементов перетяните ***Логический объект***, назовите его Calc_from_Link – калькулятор, который будет считать данные по ссылке.
2. Перейдите внутрь ***Calc_from_Link***, в панели элементов раскройте папку Ссылки и перетяните элемент ***Объектная ссылка*** на рабочую область.

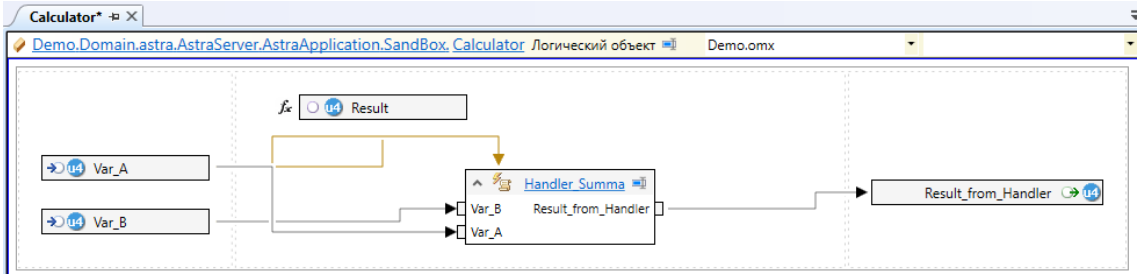


3. В свойстве Объект этой ссылки выберите Calculator, находящийся в ***AstraServer***.



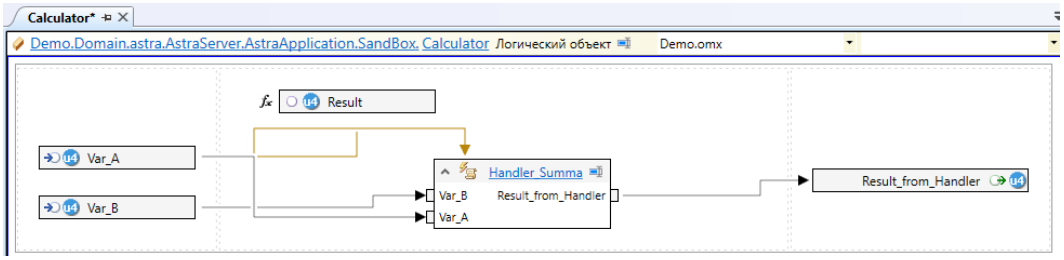
Но у этого объекта не указаны те объекты, с которыми он может взаимодействовать. То есть не указаны входные и выходные параметры. Для описания коммуникации используется такое свойство, как Направление у параметров. Если у параметра нет направления, значит, он не может участвовать в коммуникациях. Чтобы связи были доступны, нам необходимо у параметра поменять свойство Направление.

Перейдите в ***Calculator***, выделите ***переменную A*** и в свойстве Направление укажите Вход. Сделайте то же самое для ***переменной B***, а для параметра ***Result_from_Handler*** укажите направление Выход. Обновите страницу для отрисовки (F5).



Обратите внимание, теперь параметры рисуются в трех разных зонах: входная зона – слева, в середине элементы, которые не участвуют в направлении, справа – выходная зона.

4. Вернитесь в ***Calc_from_Link***, щёлкните ПКМ по ссылке и выберите **Экспонировать входы и выходы**. Студия сама создаст параметры с теми же именами и типами и нарисует соответствующие связи.



Есть ещё один способ создать ссылку на объект. Вместо того, чтобы перетягивать элемент ***Объектная ссылка*** из панели элементов, Вы можете вытянуть объект, на который хотите сослаться из Обозревателя решения.

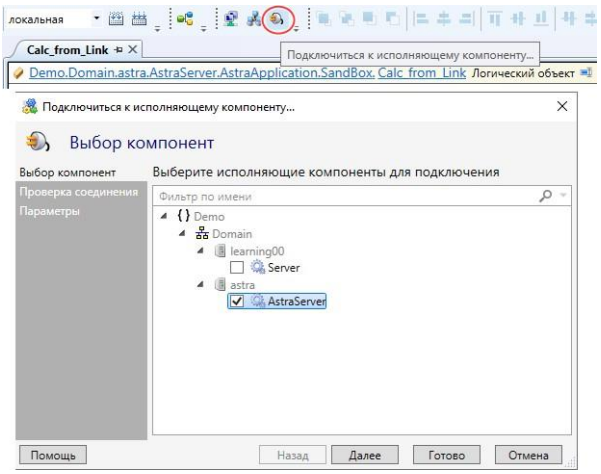
Важный момент! Из обозревателя решений перетягиваются ССЫЛКИ на объекты, а из панели элементов – ЭКЗЕМПЛЯРЫ.

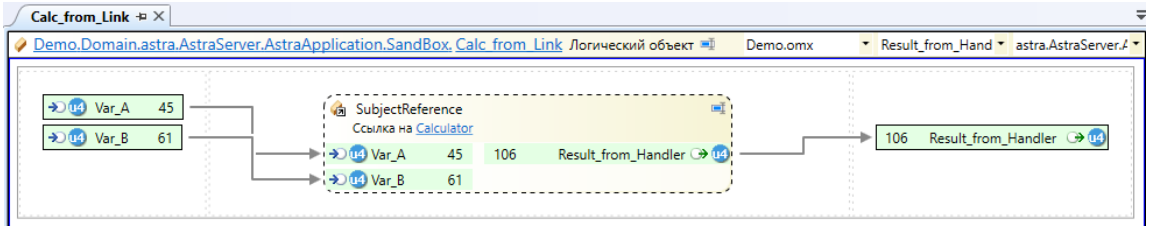
5. **Постройте решение**, **перейдите к развёртыванию** и **примените конфигурацию** к линуксовому серверу.

Теперь в проекте есть описание связи между объектами и параметрами объектов. Таким же образом можно обмениваться данными между объектами не только в рамках одного сервера, но и из внешних исполняющих компонентов. Например, с контроллерами, либо с другими серверами ввода-вывода.

Подключение к исполняющему компоненту через DeveloperStudio

DevStudio может как *OpcExplorer* напрямую подключаться к серверу. Для этого существует кнопка **Подключиться к исполняющему компоненту**. Укажите, к какому серверу подключаться (AstraServer) → нажмите Далее → дождитесь сообщения «Готов к отладке» → Далее → Поставьте галочку около Изменение значений → Готово. И *DeveloperStudio* напрямую подключится к серверу и будет получать значения.



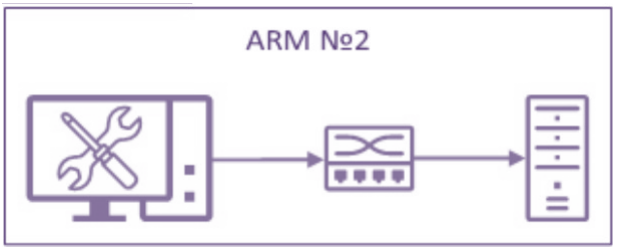


При изменении входных параметров, их значения отправляются в ***Calculator***, там происходят вычисления и результат передаётся в выходной ***Result_from_Handler*** внутри ***Calc_from_Link***.

Далее в проекте будет описание связи с внешним исполняющим компонентом – сервером ввода-вывода, имитирующего логику ПЛК.

Добавление внешних исполняющих компонентов и реализация передачи данных между машинами

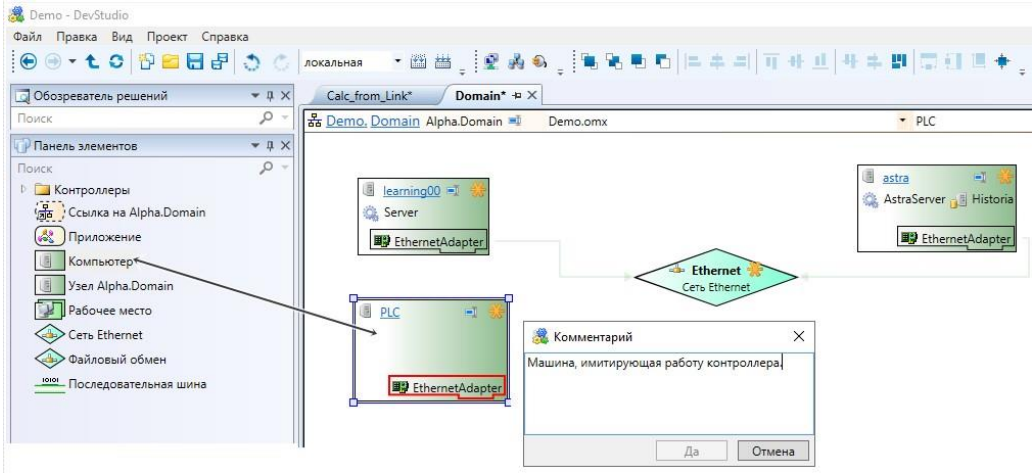
Для того, чтобы получить данные с сервера, эмулирующего работу ПЛК, необходимо на отдельной машине развернуть конфигурацию сервера. Для этого необходимо установить компоненты на вторую машину, и развернуть проект Imitator из SePlatform.DevStudio на SePlatform.Server. После развёртывания можно приступать к дальнейшему прохождению материала.



Для того, чтобы описать информационную модель обмена данными между компонентами, в проекте необходимо описать компоненты, с которыми будет происходить обмен данными, и интерфейсы обмена. В проекте мы будем описывать сервер, эмулирующий ПЛК, как внешний исполняющий компонент, находящийся на другой машине.

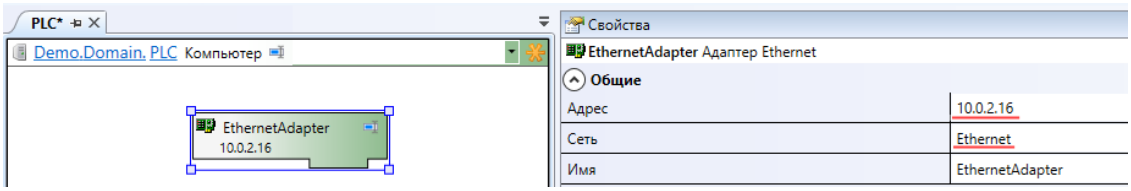
Сервер коммуницирует с устройствами, а устройства относятся к среде исполнения. Среда исполнения находится в *SePlatform.Domain*. Здесь описаны машины, то есть узлы, объединённые в одну сеть. И здесь же можно разместить какие-то внешние устройства, которые отвечают за вычислители.

1. Перейдите в ***Domain*** при помощи Обозревателя решений.
2. Перетяните сюда из панели элементов ***Компьютер***, назовите его PLC, дайте комментарий.

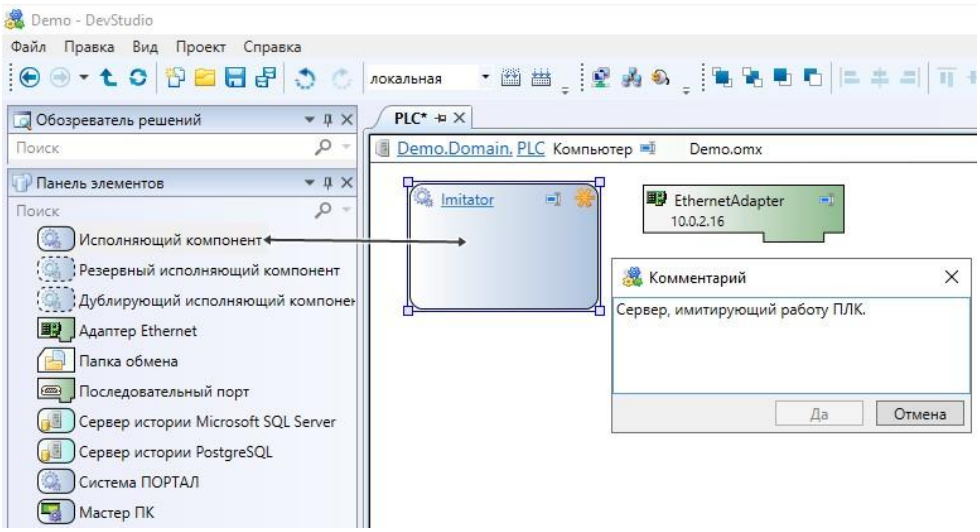


Элемент ***Компьютер*** отличается от элемента ***Узел SePlatform.Domain*** тем, что у элемента ***Компьютер*** нет свойства адрес, которое нужно для описания **SePlatform.Net.Agent** для того, чтобы определить, как идентифицировать конкретный узел в среде развёртывания.

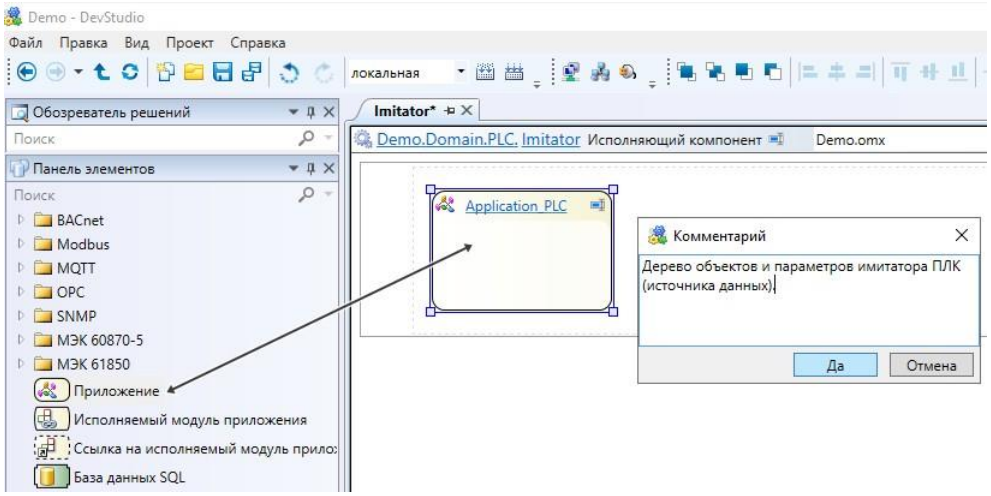
3. Перейдите в ***Компьютер PLC***, выделите ***Ethernet Adapter*** и в свойстве Адрес задайте ему IP-адрес машины, на которой развёрнут проект Imitator. Также задайте ему Сеть Ethernet.



4. Находясь внутри ***PLC*** перетяните из панели элементов ***Исполняющий компонент***, назовите его Imitator, дайте комментарий.



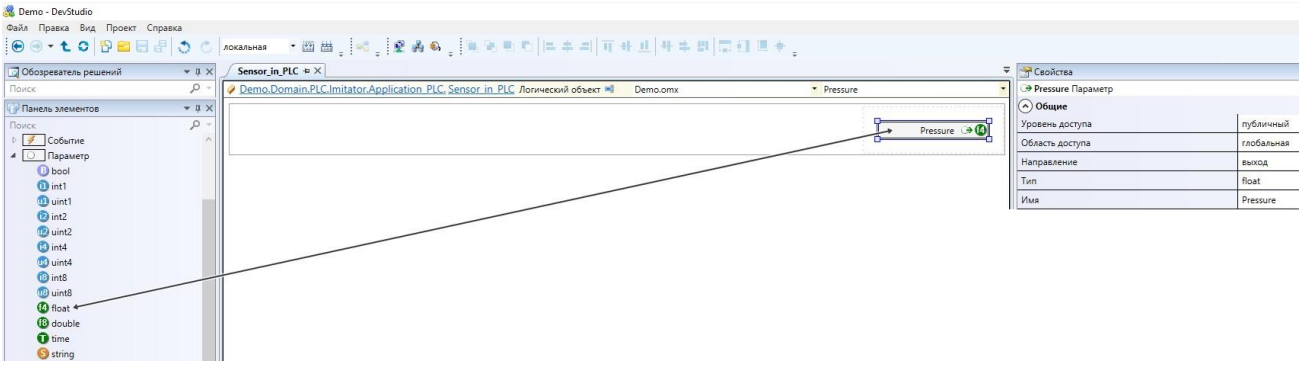
5. Для того, чтобы описать всё, что будет происходить на этом сервере, перейдите в ***Imitator*** и перетяните сюда из панели элементов ***Приложение***, назовите его Application_PLC и задайте комментарий.



Внутри этого ***Приложения*** необходимо указать объекты и параметры, с которыми будет происходить взаимодействие.

6. В проекте Imitator есть датчик, и от него мы будем получать значения. Для осуществления этого перейдите в ***Application_PLC*** и добавьте сюда ***Логический объект***, назовите его Sensor_in_PLC (датчик со стороны источника).

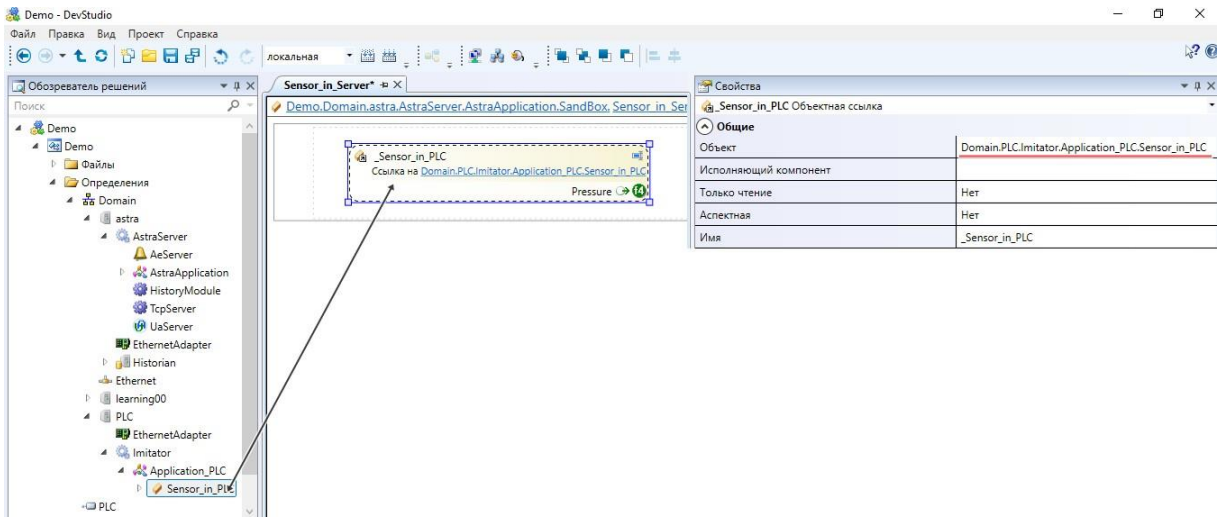
7. Перейдите внутрь ***Sensor_in_PLC*** и расположите здесь ***параметр*** типа float. Назовите его Pressure и задайте ему направление Выход.



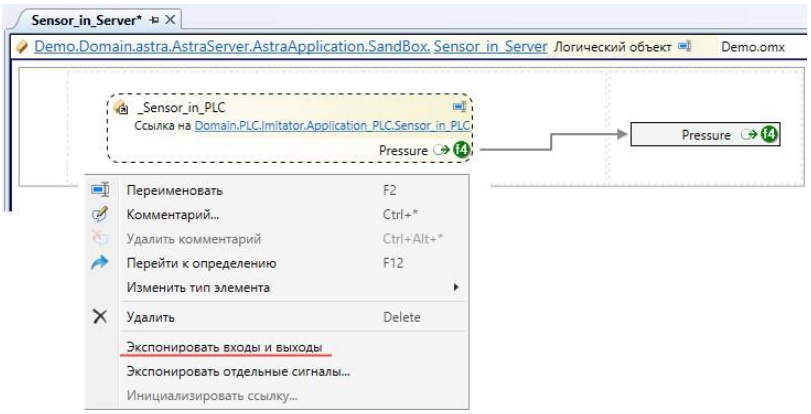
На этом описание источника с точки зрения объектно-ориентированного программирования завершено. Теперь перейдём к описанию объекта на сервере ввода-вывода (потребитель, AstraServer).

8. Перейдите в ***AstraServer*** → ***SandBox***, перетяните сюда из панели элементов ***Логический объект***, назовите его Sensor_in_Server (датчик со стороны сервера ввода-вывода).

9. Перейдите внутрь ***Sensor_in_Server*** и из Обозревателя решений перетяните сюда объект ***Sensor_in_PLC***, создавая тем самым ссылку на объект, размещенный в ***PLC***. Заметьте, что свойство Объект уже заполнено.

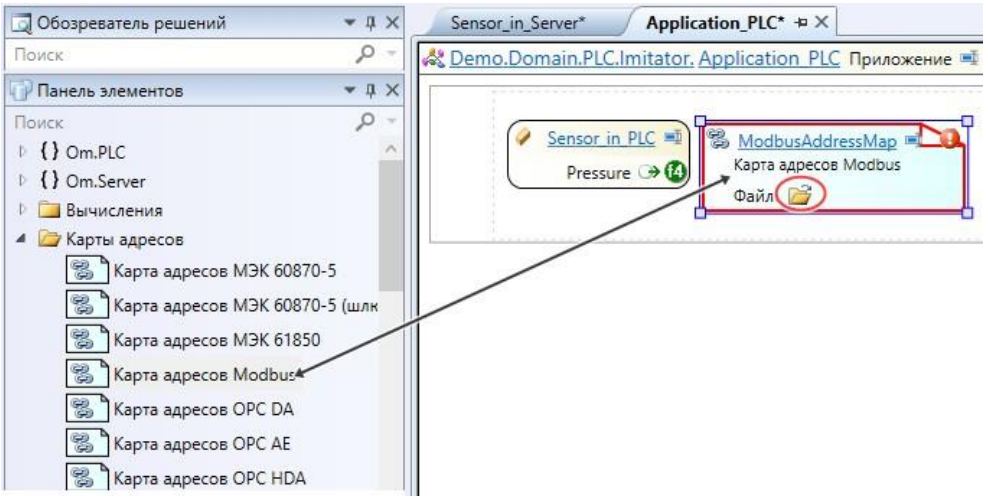


10. Кликните по ссылке ***_Sensor_in_PLC*** ПКМ → Экспонировать входы и выходы.

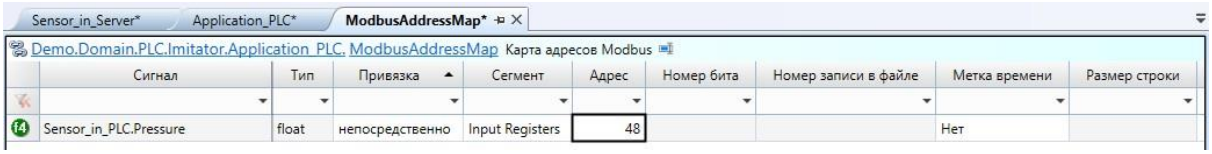


На этой объектно-ориентированный подход закончен. Осталось описать транспортный уровень.

11.Перейдите в ***Application_PLC*** и из панели элементов перетяните сюда ***Карту адресов Modbus***. Кликните на значок папки около надписи Файл и задайте имя файла MB_Map.

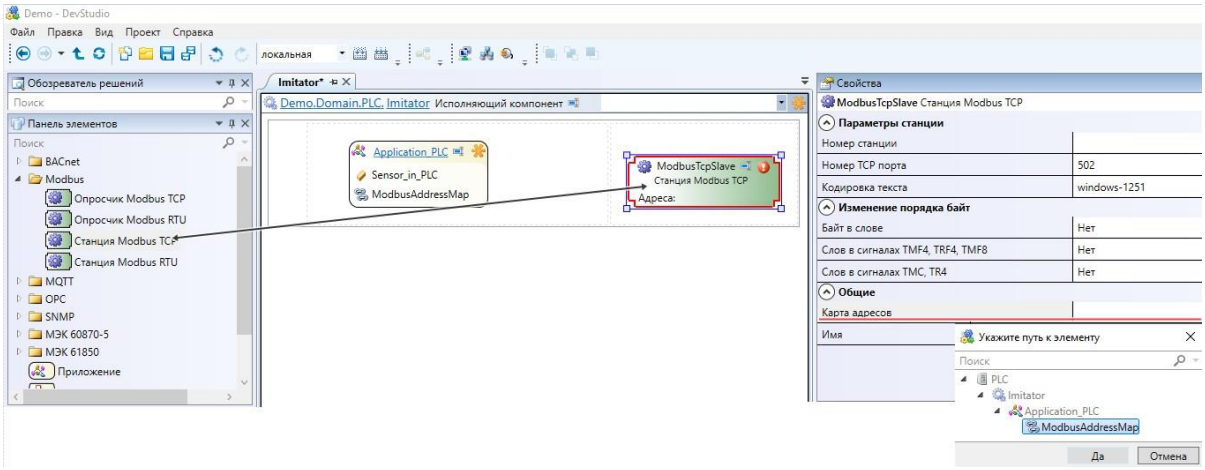


12.Зайдите внутрь ***карты адресов ModbusAddressMap*** и укажите параметры, указанные на изображении ниже (данные адреса идентичны адресам из проекта Imitator).



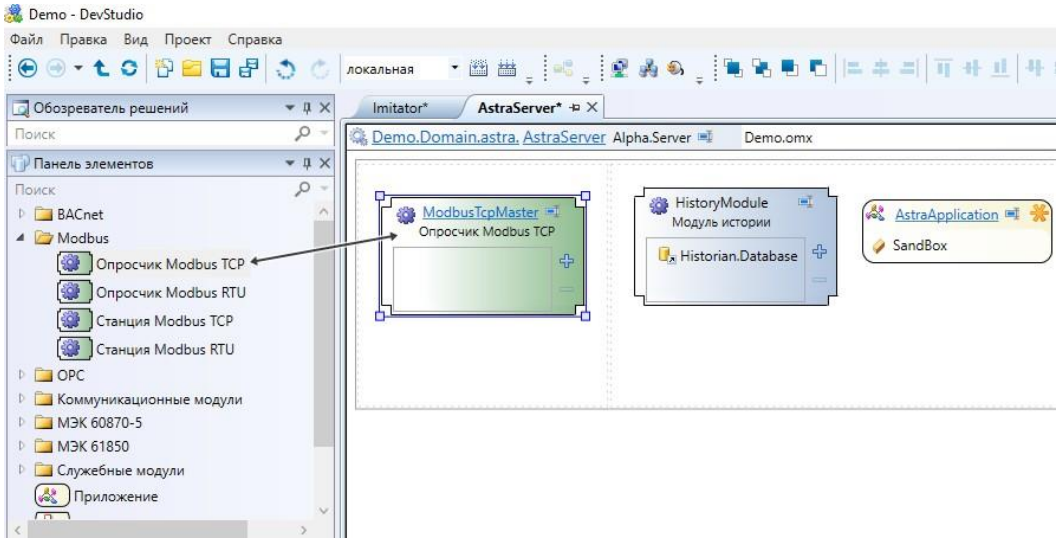
Сохраните карту и обязательно закройте её (карта обновляется при открытии).

13.Для описания того, кто эту карту адресов будет обслуживать, перейдите в ***Imitator*** и перетяните из панели элементов ***Станцию Modbus TCP***. В свойстве Карта адресов выберите ту карту ModbusAddressMap.



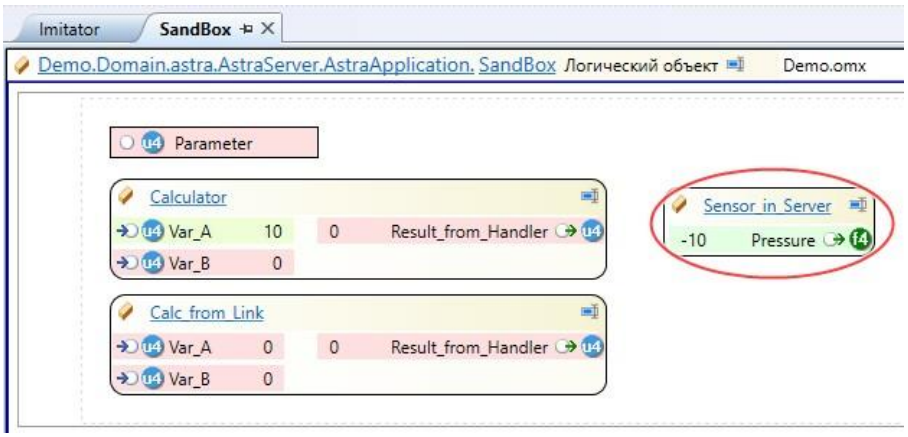
С точки зрения транспортного уровня мы сейчас описали источник. Но серверу с этим источник надо взаимодействовать. Нужно добавить модуль, который будет работать с этой станцией.

14.Перейдите в ***AstraServer*** и перетяните сюда из панели элементов ***Опросчик Modbus TCP***.



15.**Постройте решение**, **перейдите к развёртыванию** и **примените конфигурацию** к линуксовому серверу.

Для наблюдения результата перейдите в ***SandBox***.



Объектно-ориентированный подход

В *SePlatform.DevStudio* используется объектная модель, основанная на объектно-ориентированной парадигме, но имеющая отличительные особенности, связанные со спецификой области промышленной автоматизации. Основные понятия, используемые в *SePlatform.DevStudio*:

Объект

Объекты в *SePlatform.DevStudio* обычно описывают какие-либо реальные автоматизируемые объекты, либо мнимые. Объекты являются ключевой единицей информации при описании решения. Логические объекты ранее были добавлены в демонстрационный проект.

Тип

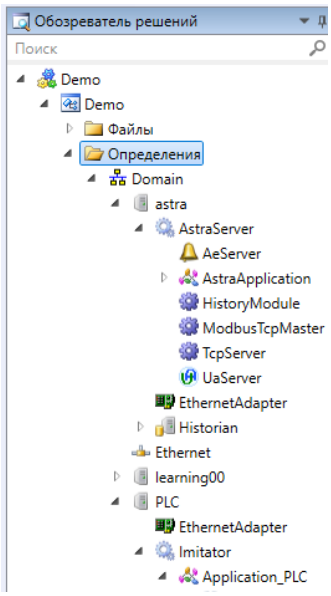
Тип (класс, шаблон) является описанием набора похожих друг на друга объектов, имеющих одинаковую роль в решении, а также общий набор данных и правил их обработки. Использование типов позволяет структурировать большое количество объектов, а также облегчает их изменение и добавление новых объектов. Далее в проекте опишем пример простого логического типа, и работу с его экземплярами.

Использование типов

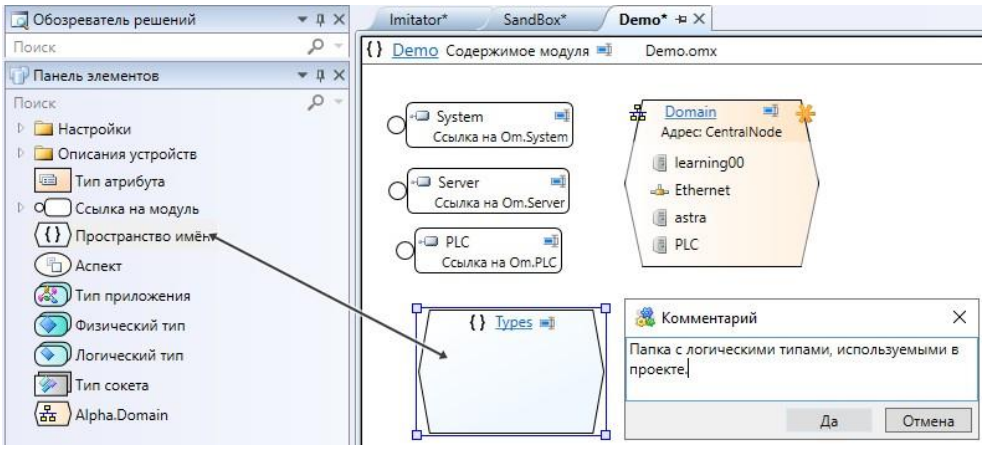
В логическом типе ***SandBox*** (песочница) есть объект ***Calculator***. Давайте попробуем создать логический тип, который будет повторять ту же самую логику, а потом его растиражируем несколько раз, чтобы понять принцип работы типизации.

Логические типы описываются за пределами среды исполнения, то есть за пределами ***Domain*** – пространство ***Определения***.

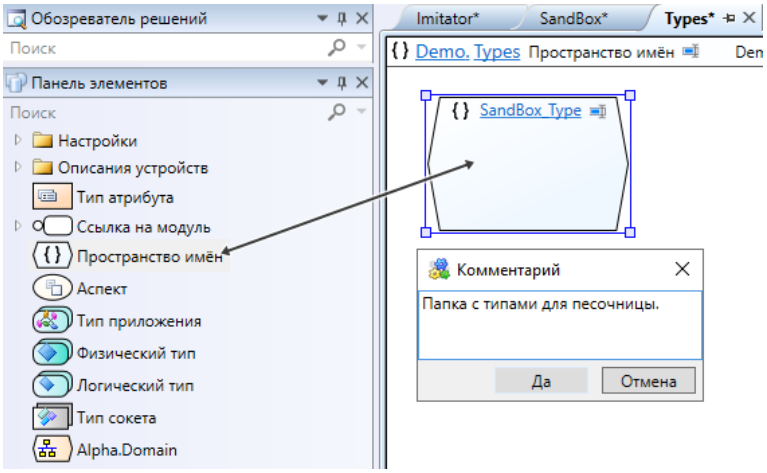
1. При помощи Обозревателя решений перейдите в пространство ***Определения***.



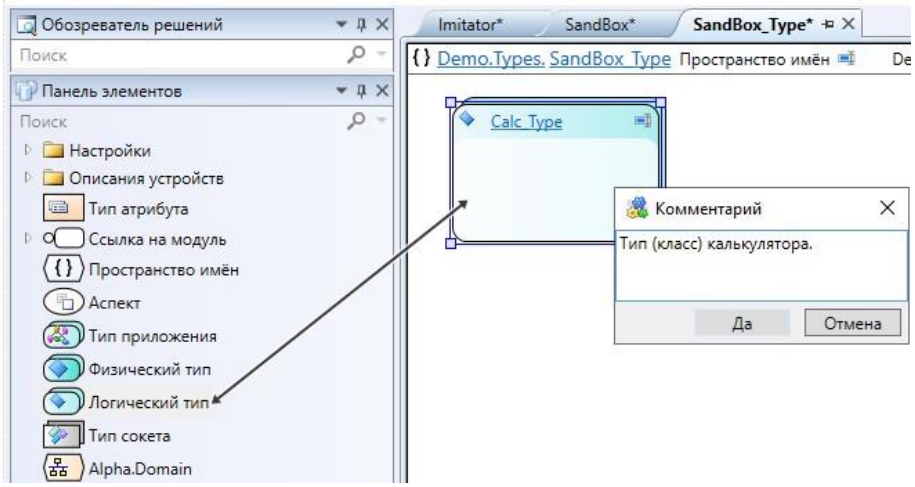
2. Перетяните в ***Определения*** из панели элементов ***Пространство имён*** (по сути это просто папка для хранения объектов). Назовите его Types, задайте комментарий.



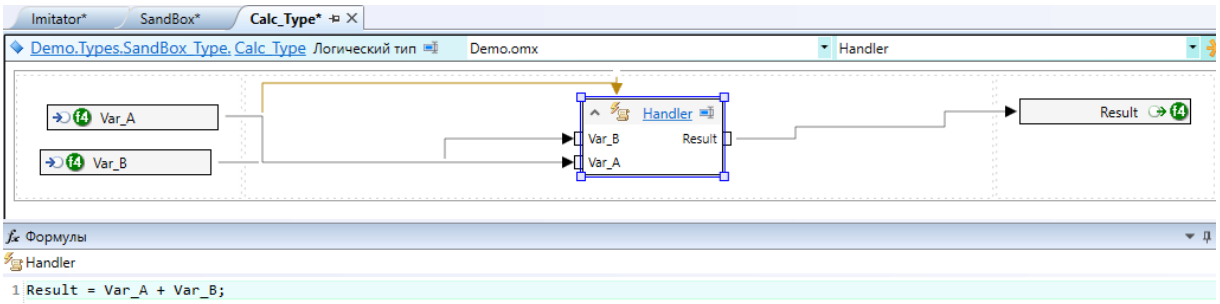
3. Перейдите внутрь ***Types*** и добавить ещё одно ***Пространство имён*** с именем SandBox_Type, задайте комментарий.



4. Перейдите внутрь ***SandBox_Type*** и из панели элементов добавьте сюда ***Логический тип***. Назовите его Calc_Type и задайте комментарий.

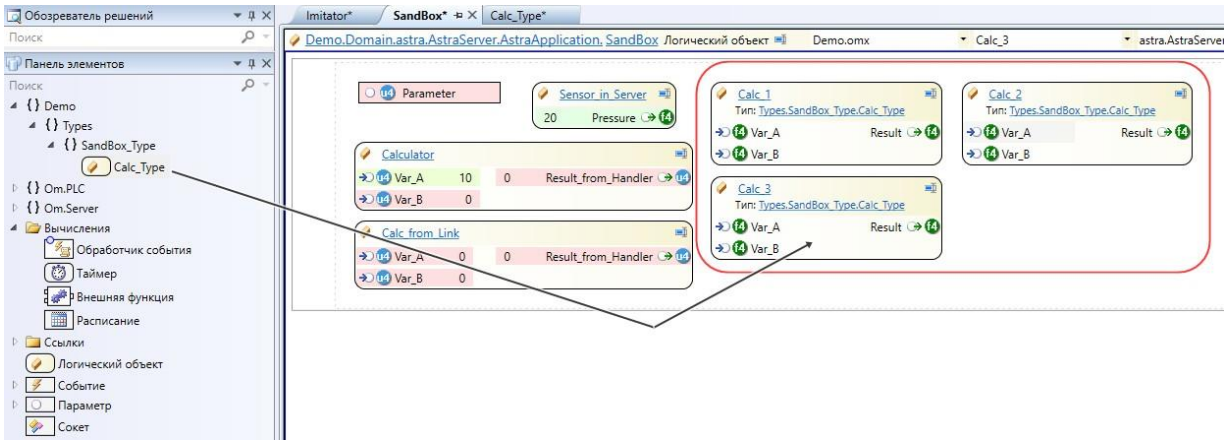


5. Перейдите внутрь ***Calc_Type*** и создайте структуру ***Calculator***, используя обработчик. Помните про направление у параметров и триггеры для обработчика.



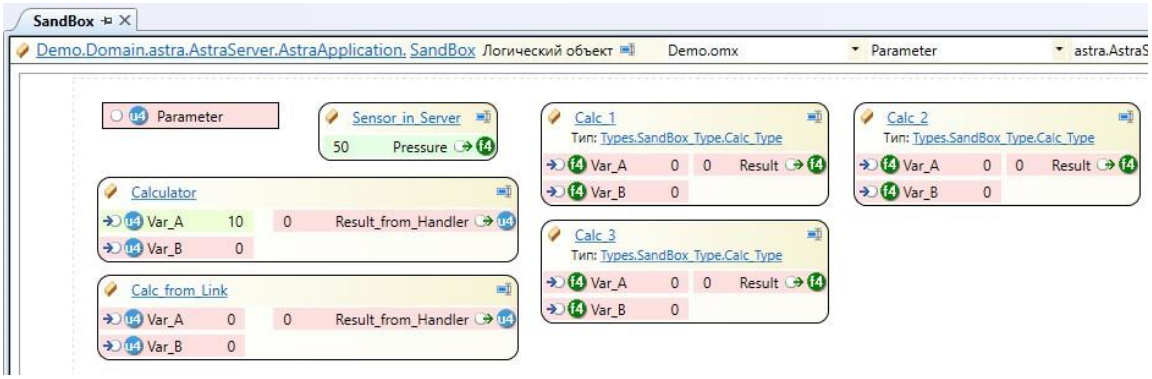
Логический тип добавлен и описан. Теперь можно приступать к тиражированию экземпляров данного типа.

6. Перейдите в ***SandBox*** на ***AstraServer*** и из панели элементов перетяните сюда 3 экземпляра ***Calc_Type***. Назовите их Calc_1, Calc_2 и Calc_3.



7. **Постройте решение**, **перейдите к развёртыванию** и **примените конфигурацию** к линуксовому серверу.

Для наблюдения результата перейдите в ***SandBox***.



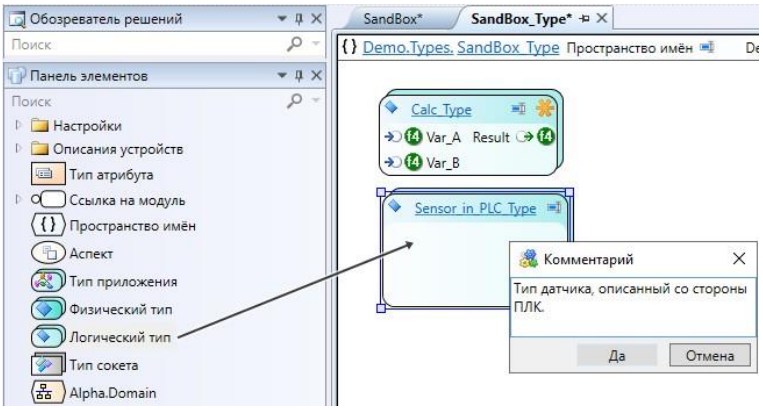
Так работает типизация. Вы один раз описали тип и много раз его переиспользовали. Экземпляры не зависят друг от друга. Логика у них одна и та же, но данные свои.

Использование ссылок в типах

Ранее в проекте описывалась передача данных от одного объекта к другому. В случае необходимости обмена данными от одного типового объекта к другому используется другая методика. Для этого в логическом типе можно описать типовую схему передачи данных от одного типа объекта к другому. Для этого в логическом типе используются ссылки и представления типов.

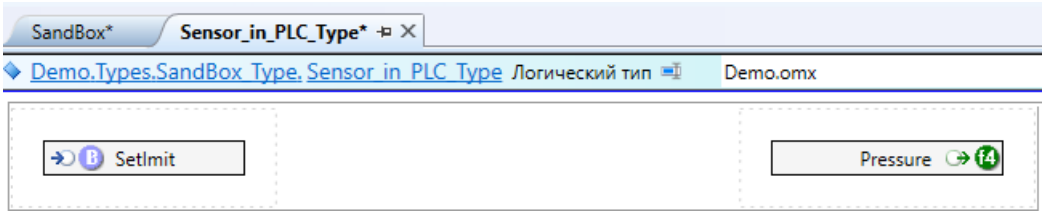
Опишем логические типы со стороны источника (PLC) и сервера ввода-вывода (AstraServer).

1. Перейдите в пространство ***Определения → Types → SandBox_Type***. Создайте здесь ***Логический тип***, назовите Sensor_in_PLC_Type и дайте комментарий.



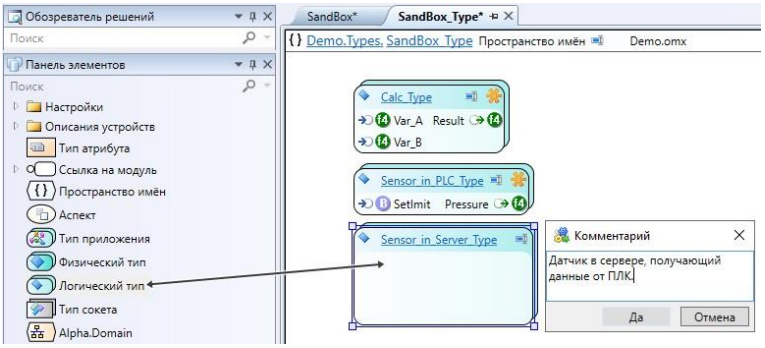
Данный датчик будет иметь возможность передавать информацию о своём давлении и может принимать команду на изменение состояния имитации.

2. Перейдите внутрь ***Sensor_in_PLC_Type***. Добавьте на вход ***параметр*** типа Bool с именем SetImit и на выход ***параметр*** типа float с именем Pressure.



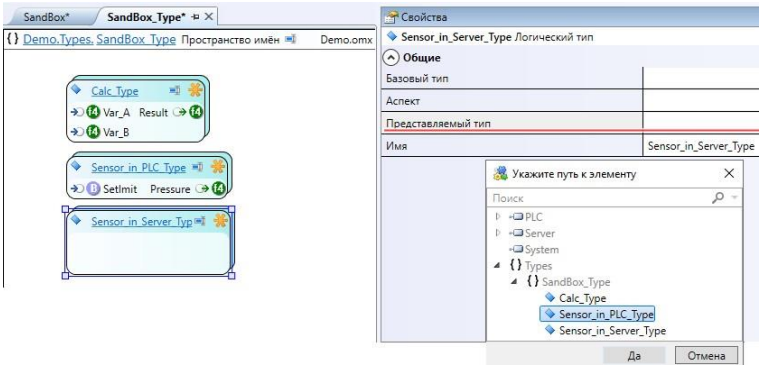
Теперь необходимо описать этот же датчик, но со стороны сервера ввода-вывода.

3. Перейдите в пространство имён ***SandBox_Type***. Создайте здесь ***Логический тип*** с именем Sensor_in_Server_Type, задайте комментарий.

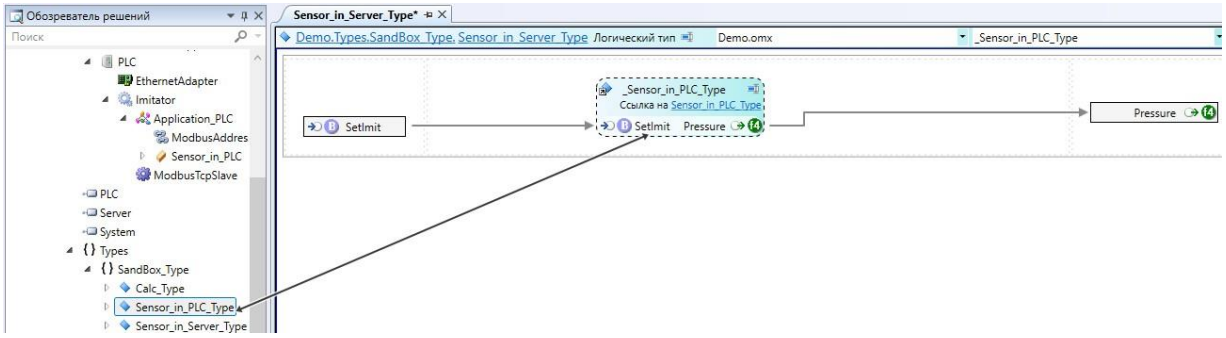


Для того, чтобы связать их между собой, необходимо указать свойство Представляемый тип.

4. Выделите ***Логический тип Sensor_in_Server_Type***, в свойстве Представляемый тип укажите Sensor_in_PLC_Type. То есть он представляет тип, описанный со стороны ПЛК.



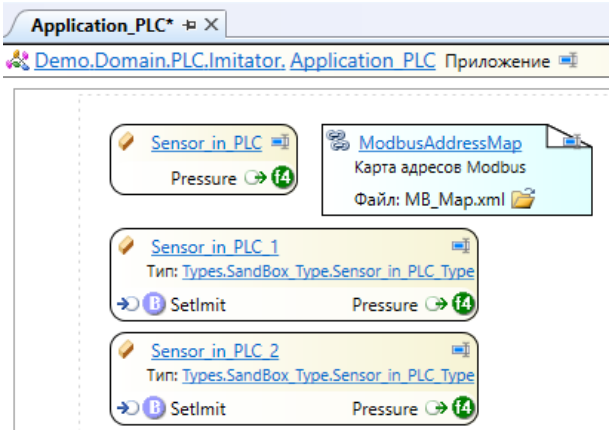
5. Перейдите внутрь типа ***Sensor_in_Server_Type***, и для того, чтобы описать взаимодействие с логическим типом, описанным со стороны ПЛК, необходимо добавить ссылку на этот логический тип. (ссылки добавляются из Обозревателя решений). Перетяните сюда из Обозревателя решений ***Sensor_in_PLC_Type***. Нажмите по этой ссылке ПКМ → Экспонировать входы и выходы.



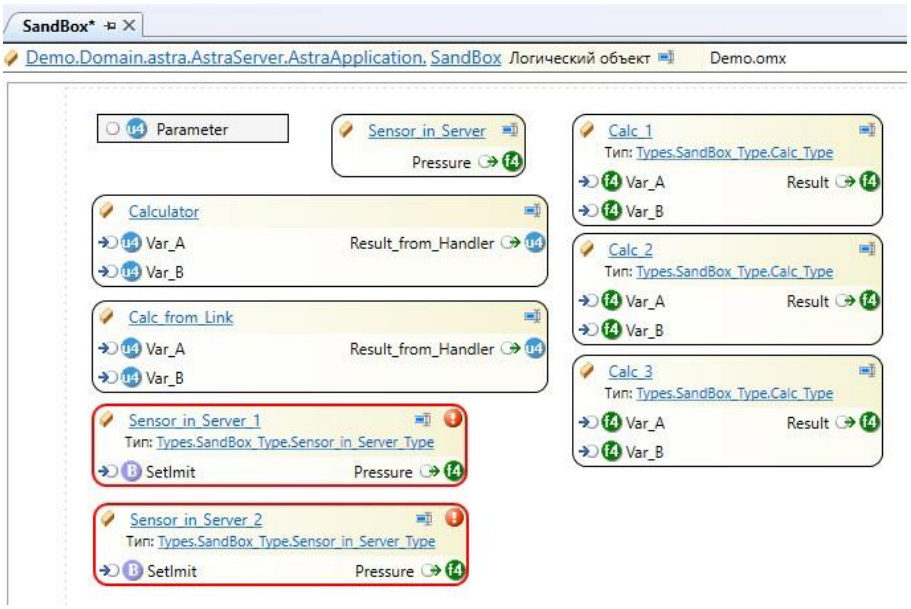
Свойство Представляемый тип гарантирует то, что в дальнейшем будет ссылка на объект, который и является экземпляром типа ***Sensor_in_PLC_Type***.

Теперь необходимо описать экземпляры в источнике и описать экземпляры в сервере.

6. Перейдите в ***Application_PLC*** и перетяните сюда из Панели элементов 2 экземпляра ***Sensor_in_PLC_Type***, назовите их Sensor_in_PLC_1 и Sensor_in_PLC_2.

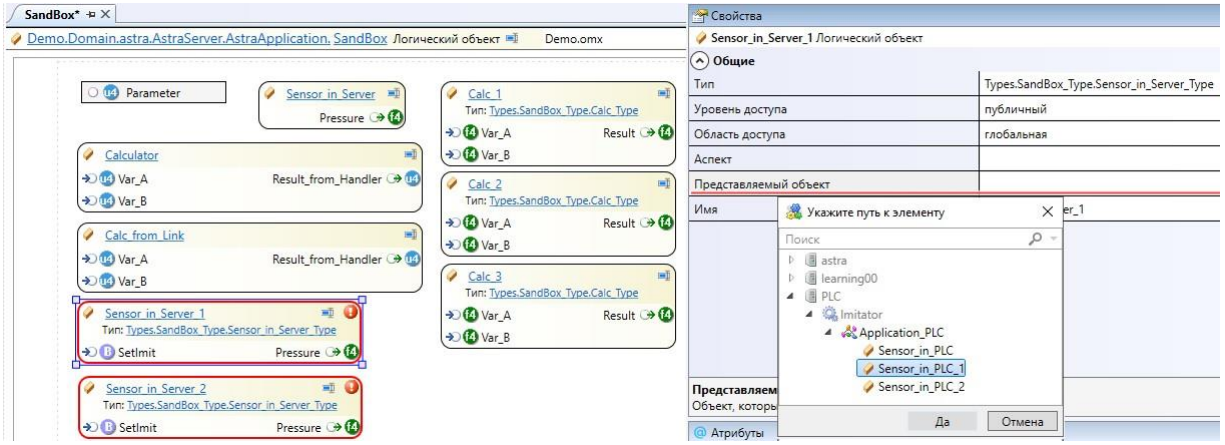


7. Перейдите в ***SandBox*** на ***AstraServer*** и перетяните сюда из Панели элементов 2 экземпляра ***Sensor_in_Server_Type***, назовите их Sensor_in_Server_1, Sensor_in_Server_2.



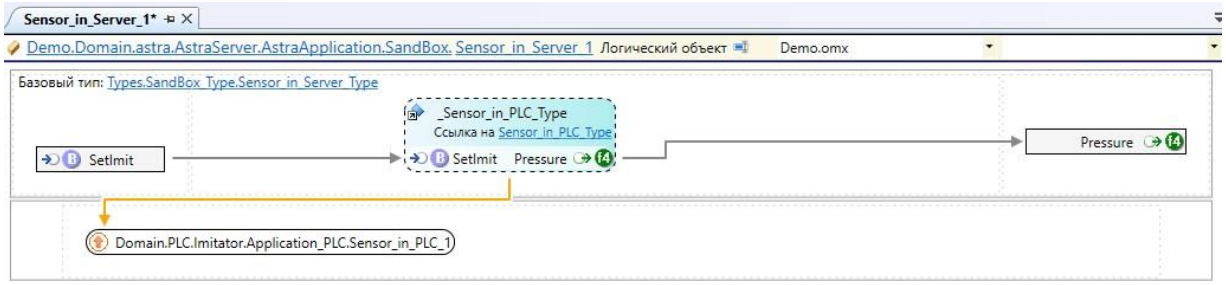
Обратите внимание, экземпляры добавляются сразу с ошибкой. Всё потому, что ссылка внутри этого объекта должна быть проинициализирована. Для этого нужно указать свойство Представляемый объект.

8. Выделите ***Sensor_in_Server_1***, в свойстве Представляемый объект укажите Sensor_in_PLC_1 (Датчик №1 на стороне сервера ввода-вывода (потребитель) представляет данные с датчика №1 со стороны ПЛК (источника)). Сделайте то же самое для ***Sensor_in_Server_2***.



9. Щёлкните ПКМ по пустому полю → Инициализировать все ссылки. Нажмите F5 для обновления страницы.

Если зайти внутрь логического объекта, ссылка ***_Sensor_in_PLC_Type***, которая вела к какому-то типу, сейчас ведёт к конкретному объекту через инициализатор. Он указывает, на какой именно объект будет вести эта ссылка. И то, что вы указали в свойстве Представляемый тип гарантирует то, что ссылка будет вести к объекту, внутри которого точно есть набор параметров SetImit и Pressure.



Теперь опишем транспортный уровень.

10.Перейдите в ***Application_PLC***, затем в ***Карту адресов ModbusAddressMap***. Заполните карту также, как на изображении ниже. Сохраните карту и закройте её.

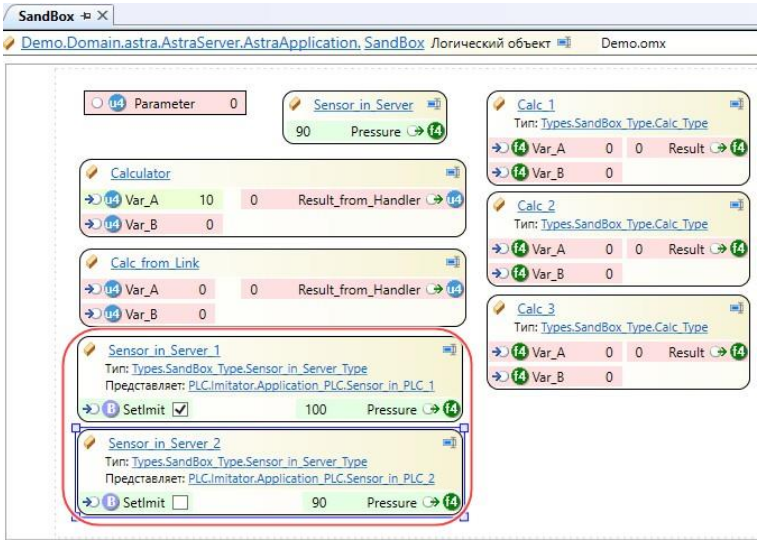
Application_PLC* | ModbusAddressMap*

Demo.Domain.PLC.Imitator.Application_PLC. ModbusAddressMap Карта адресов Modbus

Сигнал	Тип	Привязка	Сегмент	Адрес	Номер бита
Sensor_in_PLC.Pressure	float	непосредственно	Input Registers	48	
Sensor_in_PLC_1.SetImit	bool	непосредственно	Coils	7	0
Sensor_in_PLC_1.Pressure	float	непосредственно	Input Registers	50	
Sensor_in_PLC_2.SetImit	bool	непосредственно	Coils	8	0
Sensor_in_PLC_2.Pressure	float	непосредственно	Input Registers	54	

11. **Постройте решение**, **перейдите к развёртыванию** и **примените конфигурацию** к линуксовому серверу.

Для наблюдения результата перейдите в ***SandBox***. Если отправить команду SetImit = True, то имитация запустится, иначе – остановится.



Применение аспектов. Использование сокетов

Постоянно строить представления для объектов неудобно, особенно когда этих объектов много. Для удобства построения представления и описания связи на различных уровнях с помощью мастера, используются ***Аспекты***. ***Аспекты*** позволяют описать одни и те же объекты с разных сторон (точек зрения). То есть с разных уровней.

В ***DevStudio*** в решении описывается информационная модель обмена данными между компонентами. Следовательно, в проекте одни и те же объекты могут быть описаны на разных уровнях (в разных аспектах). Например, в демонстрационном проекте датчики будут описаны на среднем уровне (ПЛК) и на уровне сервера ввода-вывода (IOS). Для того, чтобы описывать объекты на разных уровнях, необходимо сначала добавить в проект уровни, чтобы в дальнейшем построить представления объектов с помощью специального мастера.

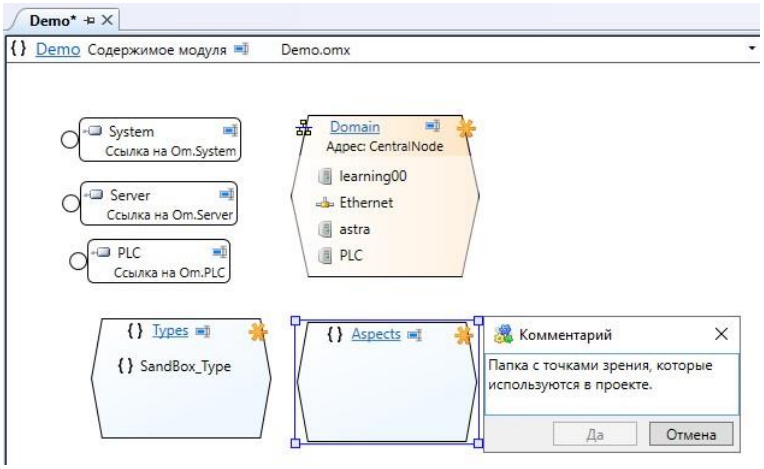
С этого момента начинается работа над демонстрационным проектом. В проекте будут описаны датчики и задвижки. Данные по датчикам будут передаваться по Modbus, а по задвижкам – по МЭК. Датчиков и задвижек будет 2 вида: простые и расширенные.

Простые датчики будут содержать на входе команду управления SetImit а на выходе сообщать о величине давления и о своём состоянии: нормальное, аварийное и предаварийное. Расширенные датчики будут содержать в себе всё то, что было у простых (наследование), а также будут дополнены уставками на чтение и на запись.

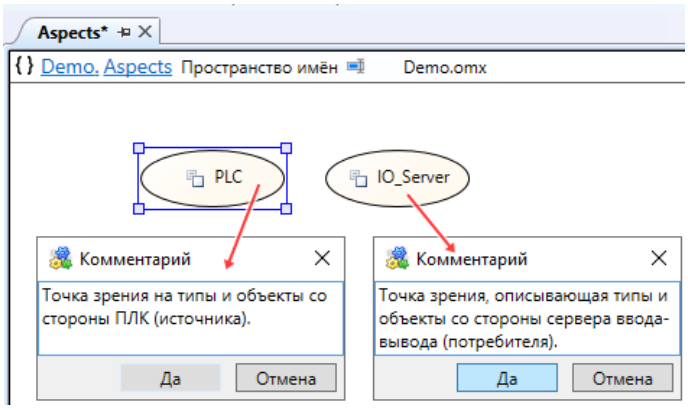
Простые задвижки на вход будут принимать команды открыть, закрыть и остановить, а на выходе будут сообщать информацию о своём состоянии (нормальное, аварийное или предаварийное) и о положении (открыта, закрыта, остановлена, в движении). Расширенные задвижки также будут наследовать структуру простых задвижек и будут дополнены параметром, характеризующим процент открытия задвижки.

В данной методичке будет описан процесс работы над созданием Простой задвижки и Расширенного датчика. Работа над частями проекта будет самостоятельной.

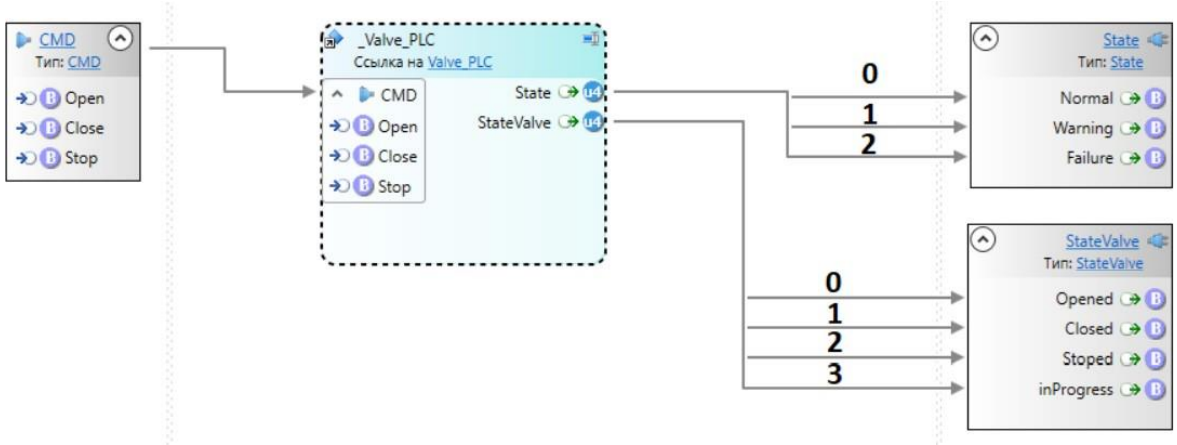
1. Перейдите в пространство ***Определения*** с помощью Обозревателя решений.
2. Перетяните сюда из Панели элементов ***Пространство имён***. Назовите его Aspects, задайте комментарий.



3. Перейдите внутрь ***Aspects*** и перетяните сюда из панели элементов 2 экземпляра элемента ***Аспект***. Назовите их PLC и IO_Server, задайте комментарий.



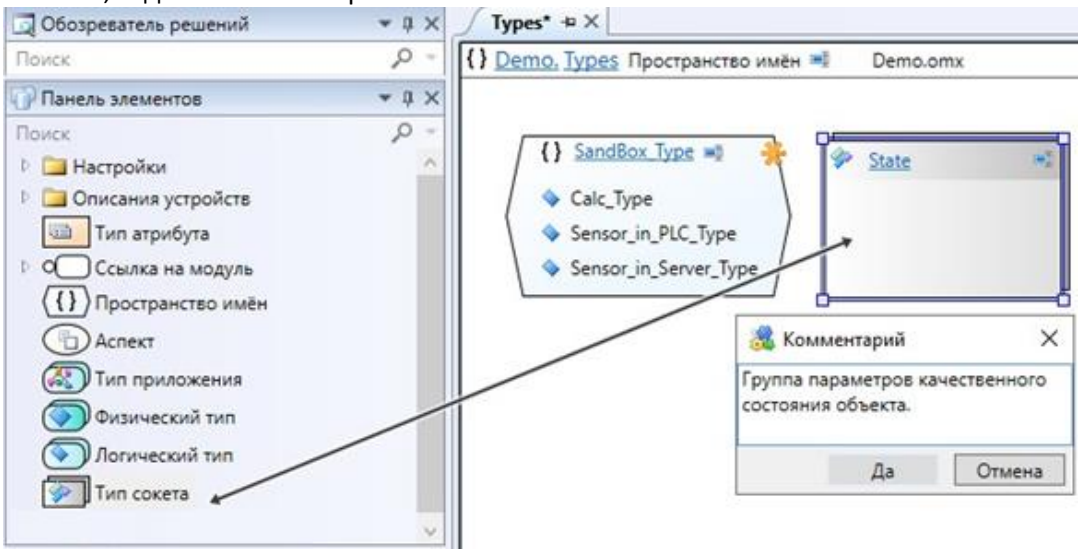
Начнём создание Простой задвижки. Её структура представлена на изображении ниже.



Как можно заметить, команды, положения и состояния сгруппированы в один элемент – сокет. ***Тип сокета*** – это группа параметров, которую можно использовать сколь угодно раз и где угодно.

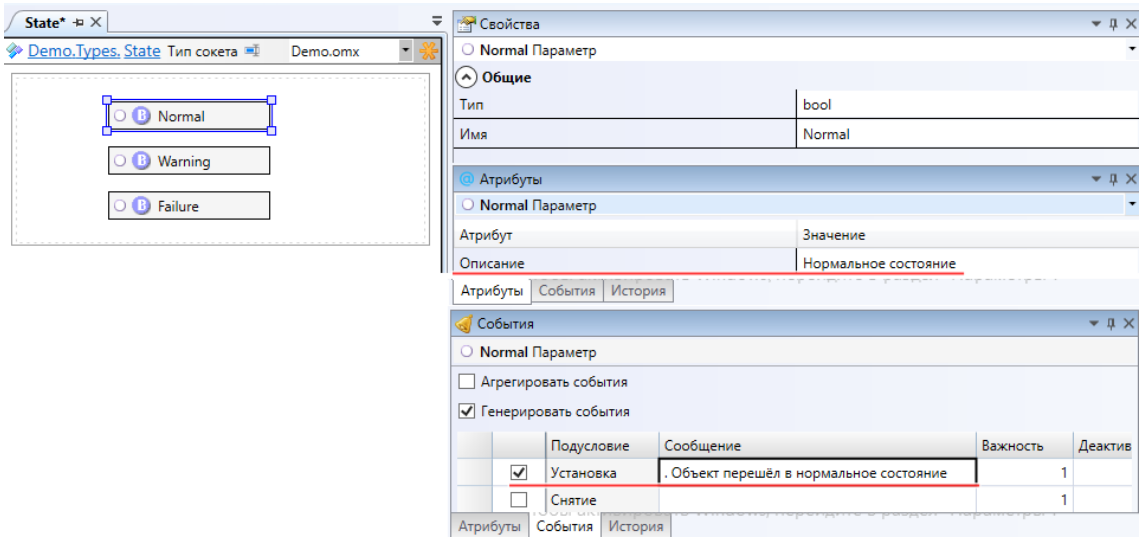
Комментарий: сокет State общий для датчиков и задвижек.

4. Перейдите в пространство имён ***Types*** и перетяните сюда из Панели элементов ***Тип сокета***. Назовите его State, задайте комментарий.

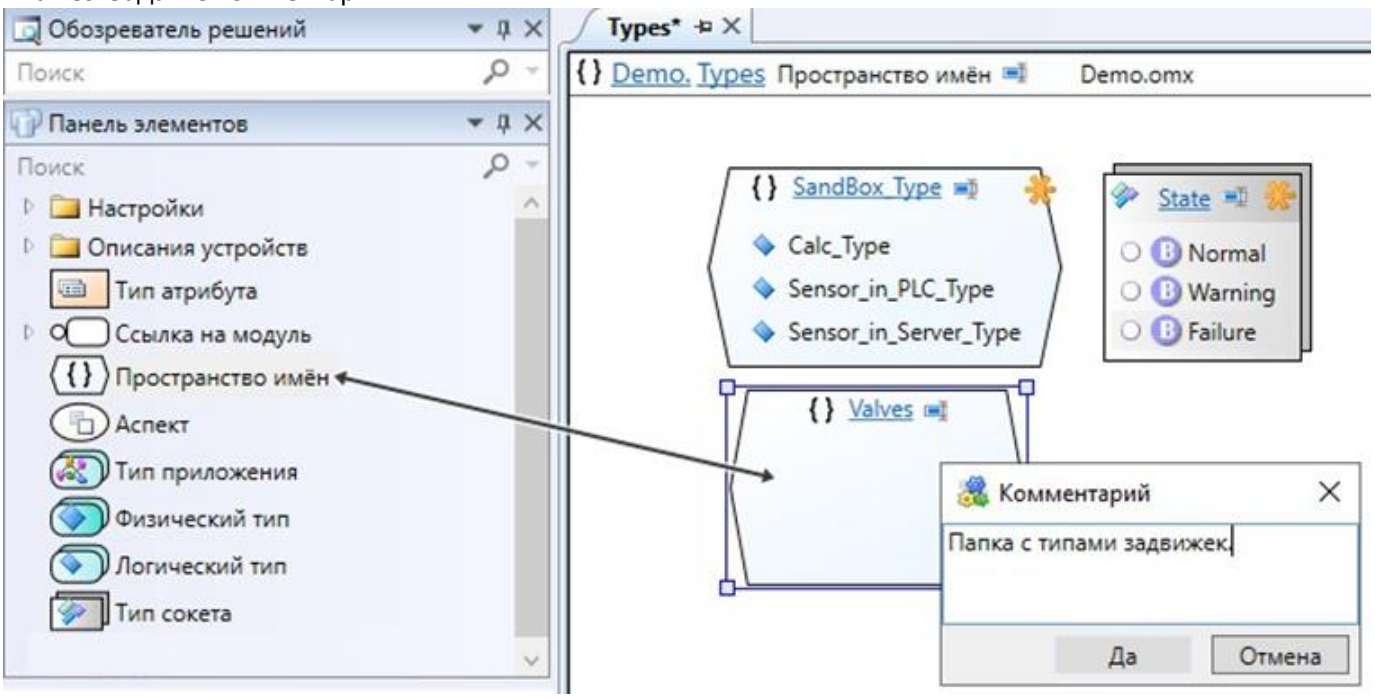


5. Перейдите внутрь ***сокета State*** и добавьте сюда 3 ***параметра*** типа Bool: Normal, Warning и Failure.

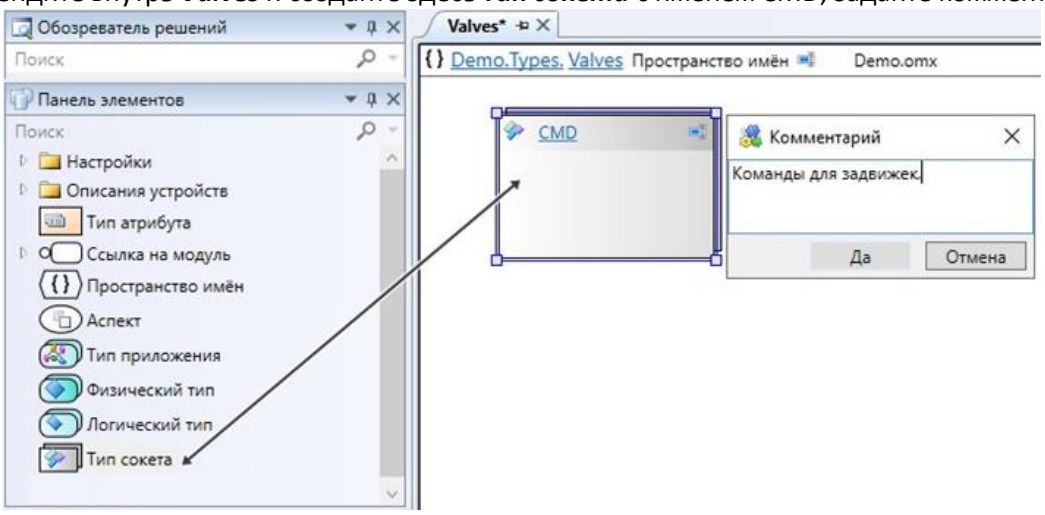
6. Добавьте каждому из параметров атрибут Описания (нормальное состояние, предаварийное состояние, аварийное состояние) и генерацию события по установке (Объект перешёл в нормальное, предаварийное или аварийное состояние).



7. Вернитесь в Пространство имён ***Types*** и создайте здесь новое ***Пространство имён*** с именем Valves. Задайте комментарий.

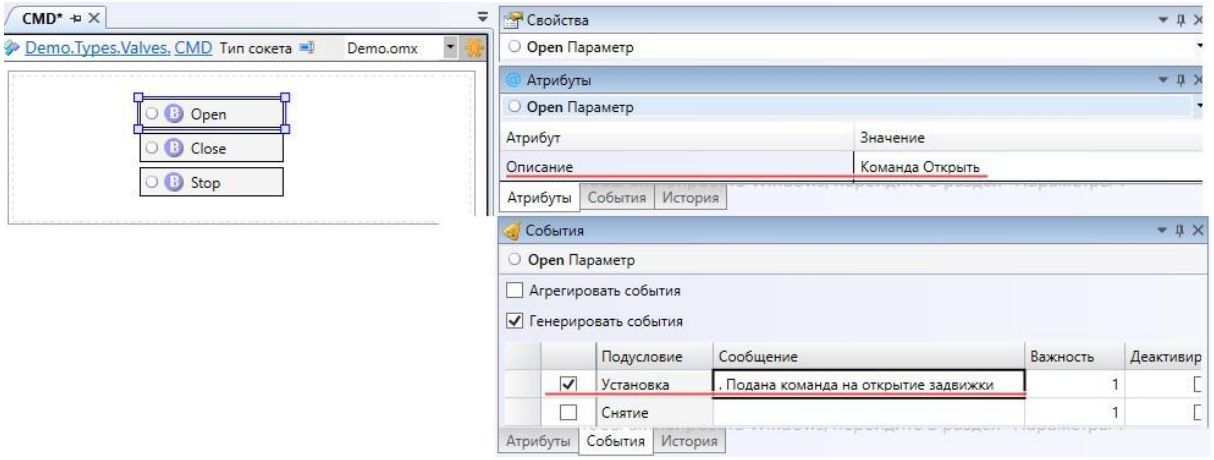


8. Перейдите внутрь ***Valves*** и создайте здесь ***Тип сокета*** с именем CMD, задайте комментарий.

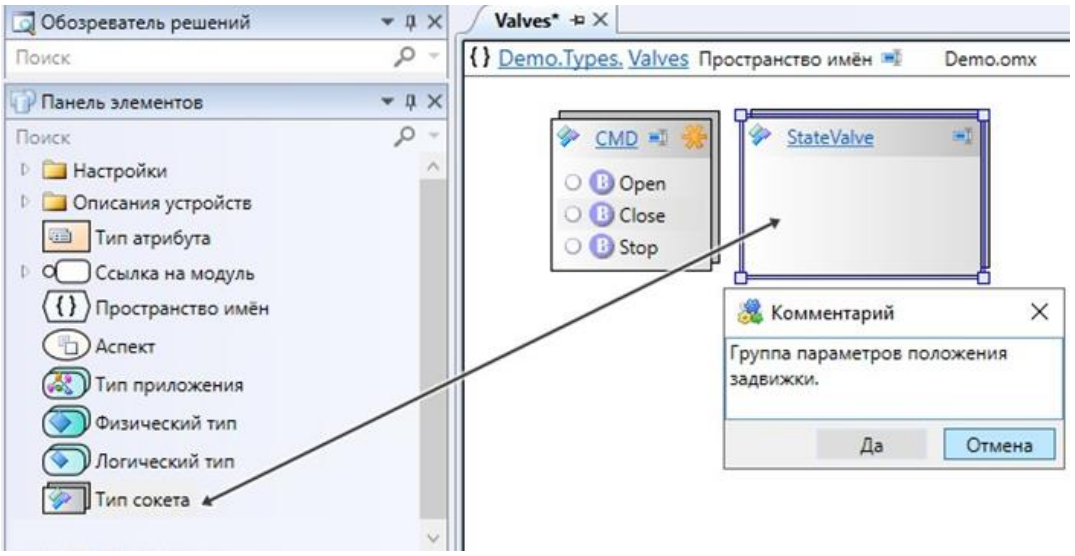


9. Перейдите внутрь ***сокета CMD*** и перетяните сюда из Панели элементов 3 экземпляра ***параметра*** типа Bool. Назовите их Open, Close и Stop.

10.Добавьте каждому из ***параметров*** атрибут Описания (команда открыть, закрыть, остановить) и генерацию события по установке (подана команда на открытие, закрытие и остановку задвижки).

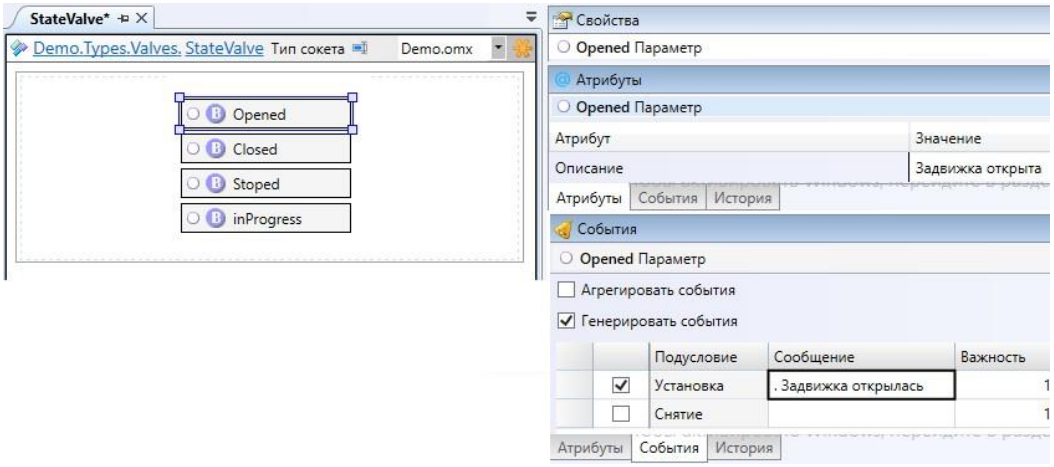


11.Вернитесь в Пространство имён ***Valves*** и создайте ещё один ***Тип сокета***. Назовите его StateValve и задайте описание.



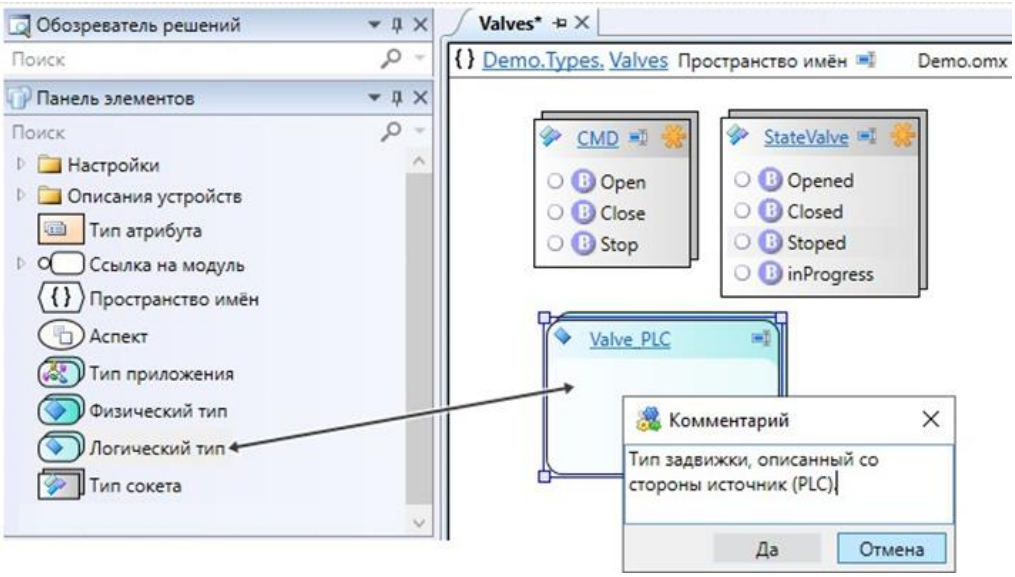
12.Зайдите внутрь ***Типа сокета StateValve*** и добавьте сюда 3 экземпляра ***параметра*** типа Bool. Назовите их Opened, Closed, Stoped и inProgress.

13.Добавьте каждому из ***параметров*** атрибут Описания (задвижка открыта, закрыта, остановлена и в движении) и генерацию события по установке (задвижка открылась, закрылась, остановилась и пришла в движение).

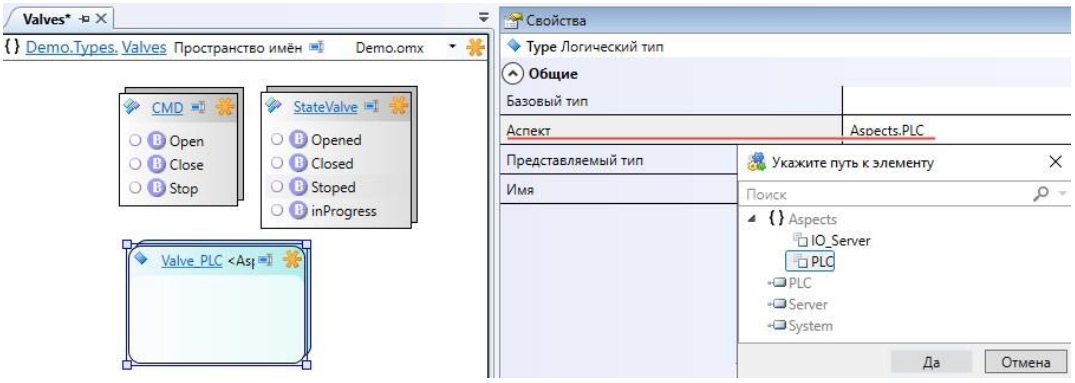


Теперь перейдём к созданию логических типов. Сначала опишем логический тип со стороны источника (PLC), а после – со стороны потребителя (IO_Server).

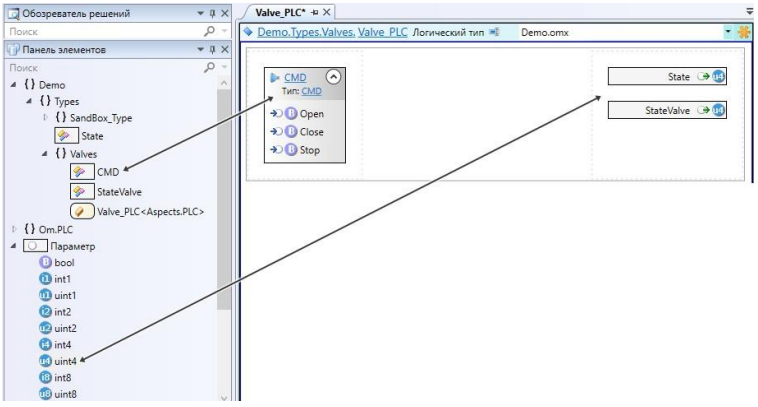
14.Перейдите в Пространство имён ***Valves*** и создайте здесь ***Логический тип***. Назовите его Valve_PLC, задайте комментарий.



15.Выделите ***Логический тип Valve_PLC*** и в свойстве Аспект укажите PLC (точка зрения источника).

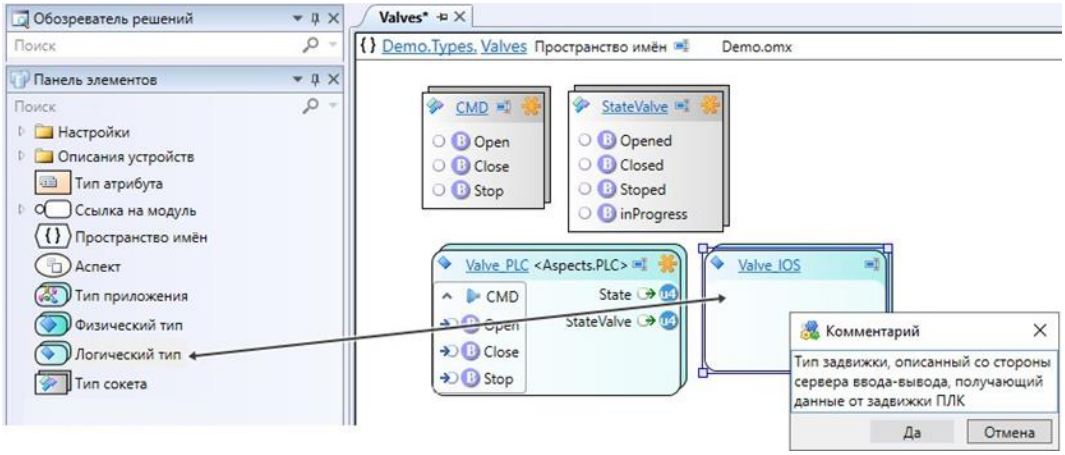


16.Перейдите в Логический тип ***Valve_PLC*** и перетяните сюда из Панели элементов на вход ***сокет CMD***, а на выход 2 ***параметра*** типа Uint4. Назовите их State и StateValve.

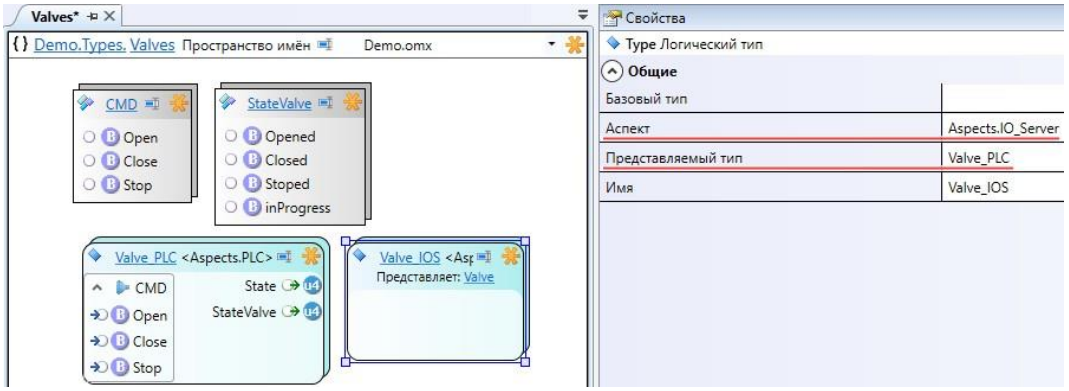


17.Перейдите в ***Пространство имён Valves*** и создайте здесь ***Логический тип***. Назовите его Valve_IOS, задайте комментарий.

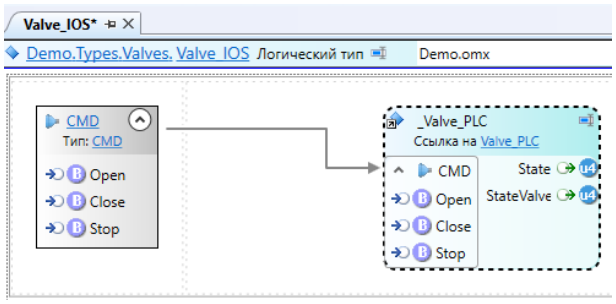
18.



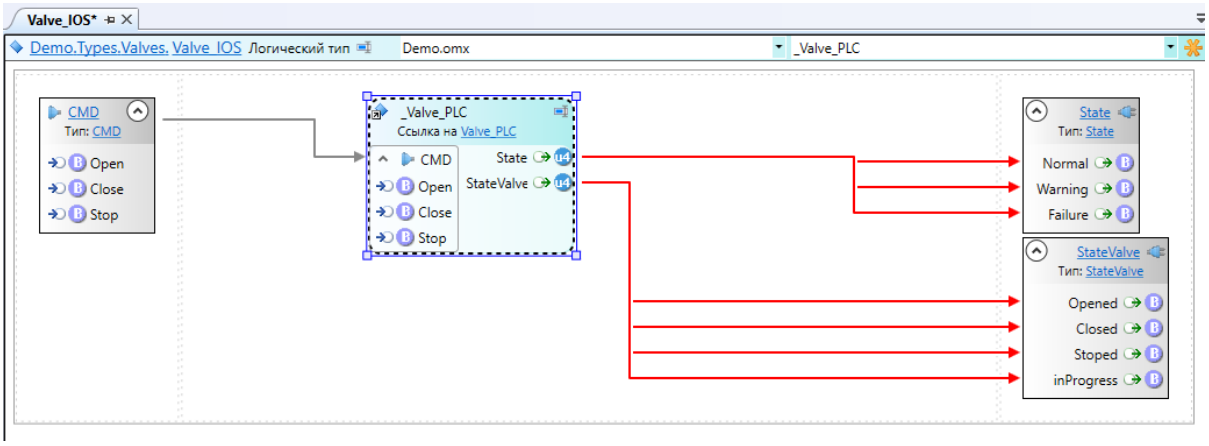
19.Выделите ***Логический тип Valve_IOS***, в свойстве Аспект укажите IO_Server, в свойстве Представляемый тип – Valve_PLC (задвижка со стороны сервера ввода-вывода (потребителя) представляет данные с задвижки, описанной со стороны ПЛК (источника)).



20.Зайдите внутрь ***Логического типа Valve_IOS***. Для того, чтобы описать взаимодействие с типом со стороны ПЛК, необходимо добавить ссылку, но теперь ссылку будем добавлять немного иначе. Щёлкните ПКМ по пустому рабочему полю → Добавить → Аспектную ссылку. Выберите представление, на которое необходимо добавить ссылку – Valve_PLC → Далее → Далее. Выберите элементы ссылки, которые требуется экспонировать – Сокет CMD → Готово.

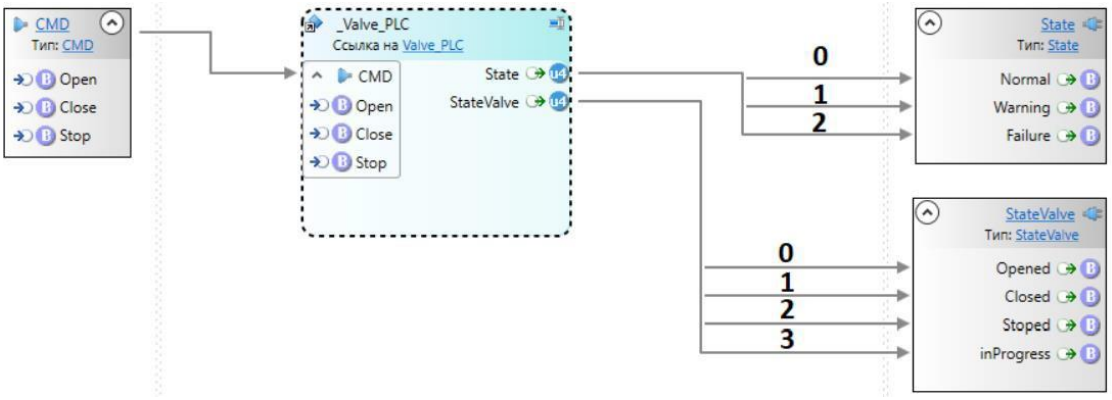


21.***State*** и ***StateValve*** будем разбивать на булевский параметры. Для этого из Панели элементов на выход перетяните ***сокеты State*** и ***StateValve***. Теперь от ***параметра State*** типа Uint4 протяните связи к каждому булевскому параметру из ***сокета State***. То же самое сделайте и для ***параметров сокета StateValve***.



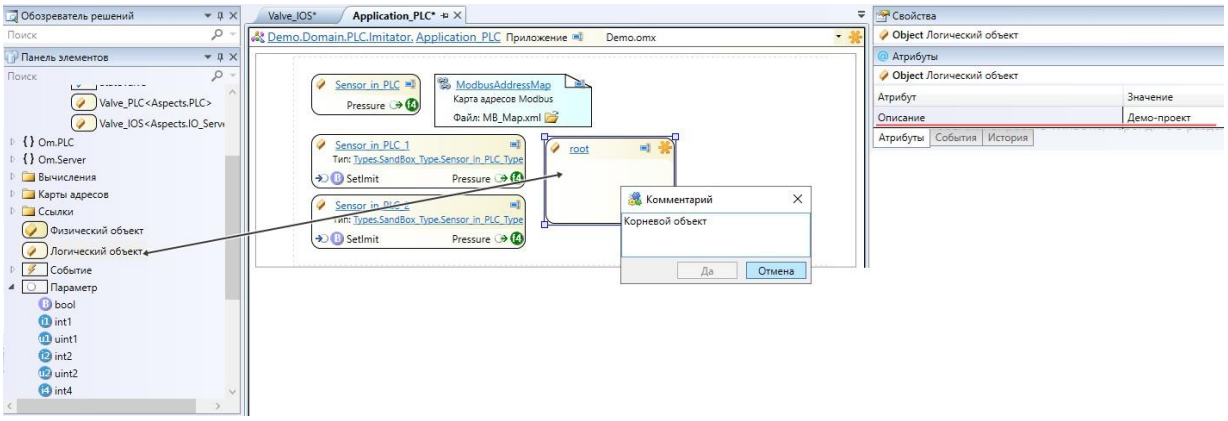
DevStudio ругается на эту связь, так как нельзя из целочисленного параметра передать информацию булевскому параметру. Чтобы это исправить, необходимо каждую из этих связей преобразовать.

22.Выделите по очереди каждую из связей и в свойстве Преобразование установите Получить бит. Свойство Номер бита каждой связи укажите в соответствии с изображением.

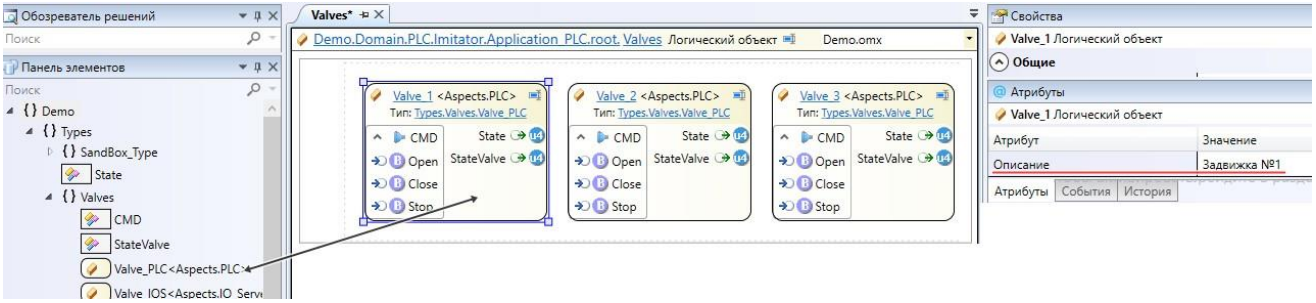


Объектно-ориентированный подход с точки зрения типизации описан. Теперь необходимо добавить экземпляры со стороны ПЛК и со стороны сервера ввода-вывода, т.е. сначала источник, потом потребитель.

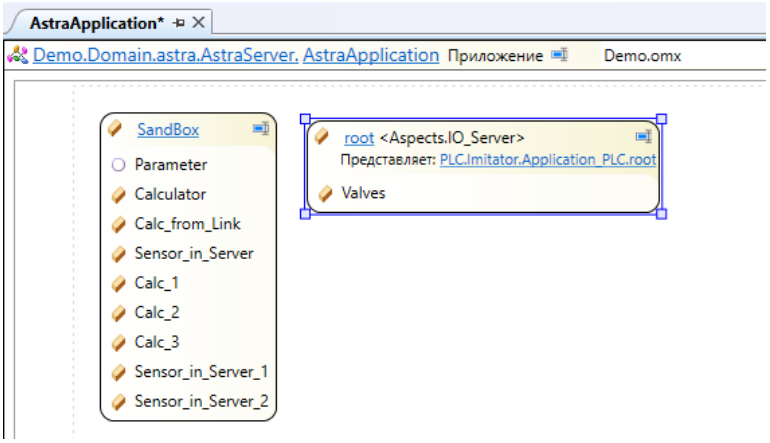
23. Перейдите в ***Aplication_PLC***. Добавьте сюда ***Логический объект***, назовите его root. Добавьте ему комментарий и атрибут Описание.



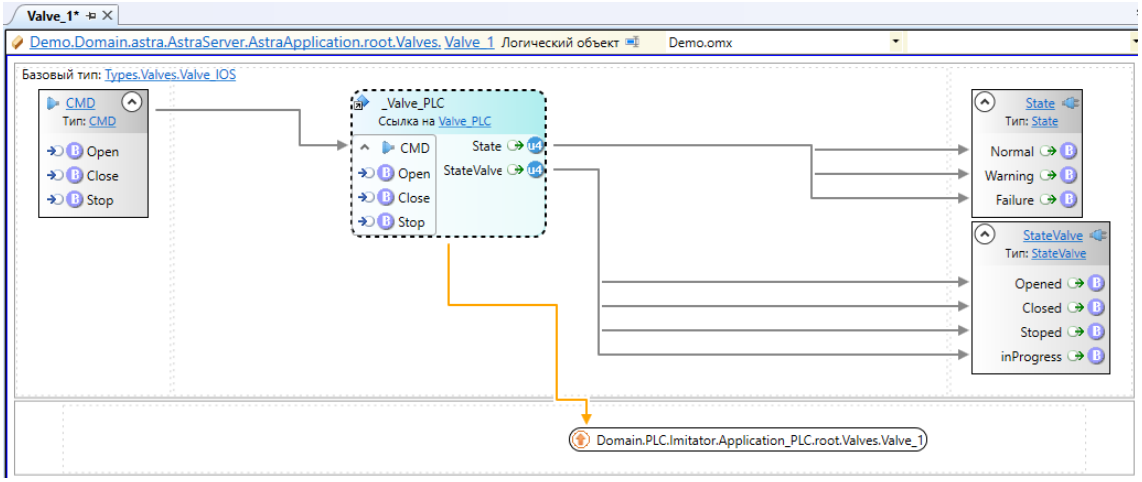
24. Зайдите внутрь ***root*** и создайте здесь ***Логический объект***. Назовите его Valve и добавьте ему атрибут Описание, в нём запишите Задвижки.
25. Зайдите внутрь ***root*** и перетяните сюда из Панели элементов 3 экземпляра типа ***Valve_PLC***. Назовите их Valve_1, Valve_2, Valve_3. Задайте каждому из экземпляров атрибут описания (Задвижка №1, Задвижка №2, Задвижка №3).



26. Перейдите в ***AstraApplication***, кликните ПКМ по пустому рабочему полю → Создать здесь представления объектов. Выберите полностью объект root из PLC → Далее → Далее → Укажите аспект новых представлений (Aspects.IO_Server, так как вы находитесь внутри сервера ввода-вывода) → Готово.

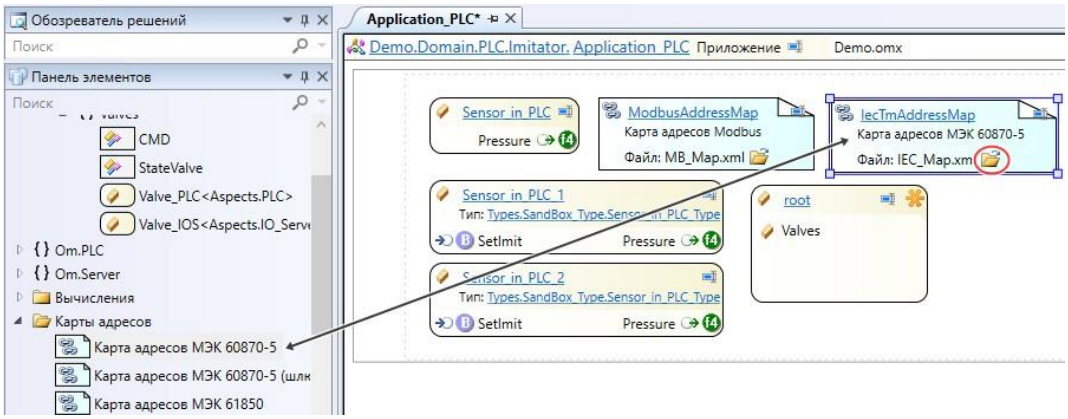


Если зайти внутрь любой из задвижек со стороны сервера ввода-вывода, то можно увидеть, что ссылка внутри объекта проинициализирована (ведёт к конкретному объекту со стороны ПЛК, связанному с ним одним представлением).

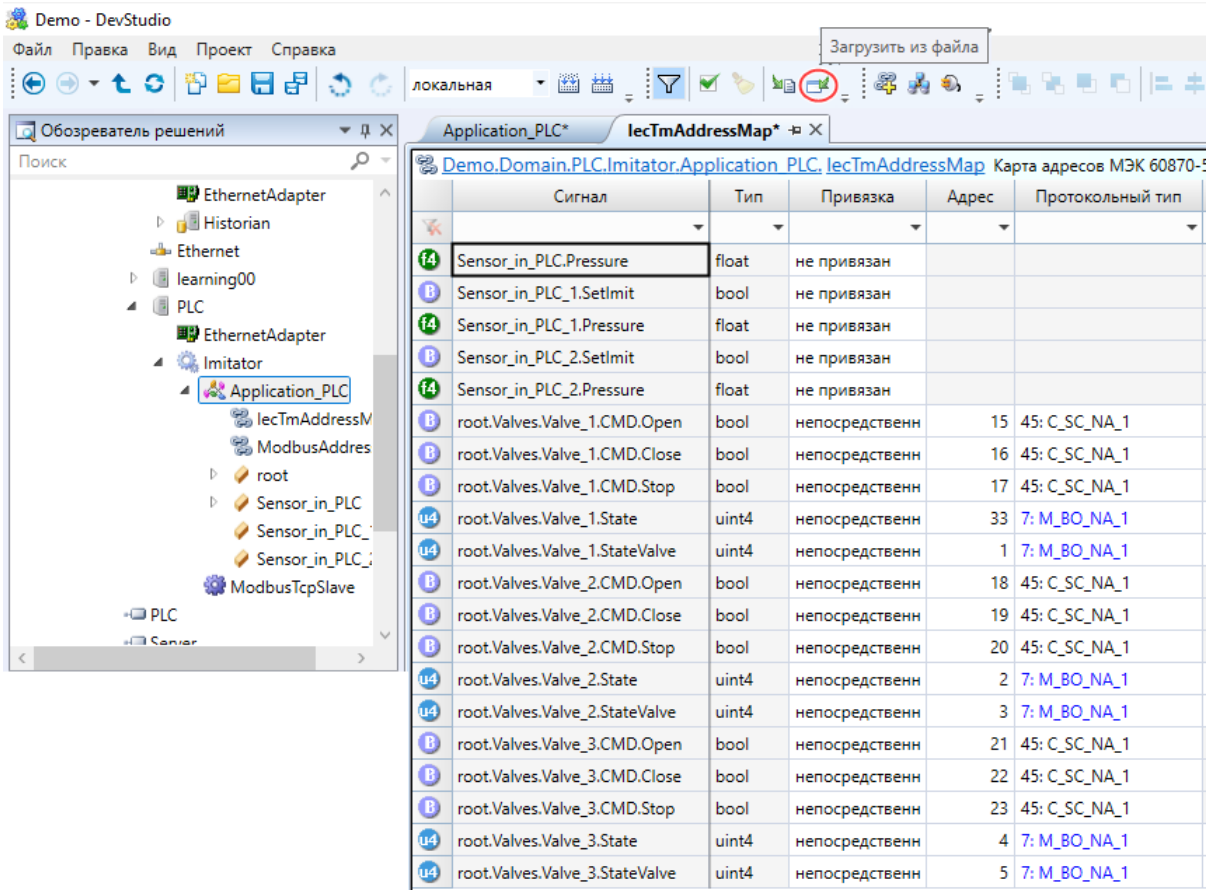


Объектно-ориентированный подход описан, теперь необходимо описать транспортный уровень. Информация о задвижках передаётся по 104 МЭКу.

27. Перейдите в ***Application_PLC***, перетяните сюда из Панели элементов ***Карту адресов МЭК***. Нажмите на файлик внутри и создайте файл с именем IEC_Map.

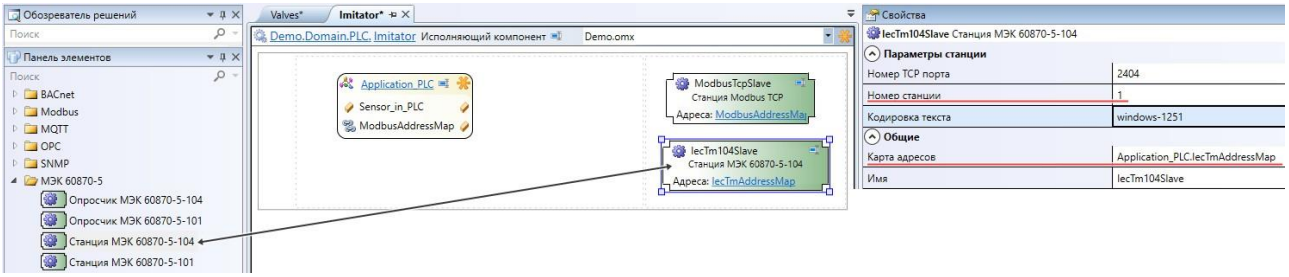


28. Зайдите внутрь ***Карты адресов IecTmAddressMap***. Для заполнения карты адресов воспользуйтесь уже готовым файлом, содержащим в себе все адреса. Для этого нажмите на кнопку **Загрузить из файла** и выберите файл iec_export_Map_Valves.xlsx. Если имена всех параметров и объектом совпадают с тем, как описано в методичке, то все необходимые адреса импортируются. Сохраните карту и закройте её.

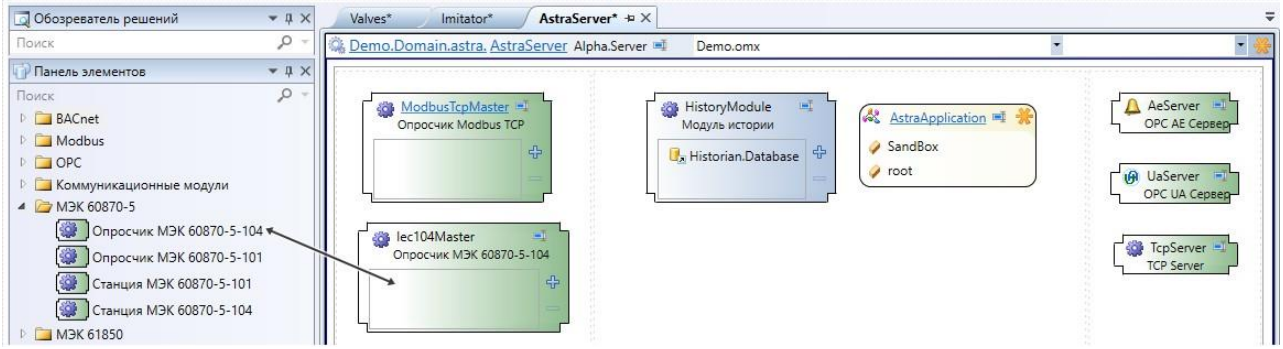


Раз появилась новая карта адресов, необходимы также соответствующие станция и опросчик.

29. Поднимитесь на уровень выше (в ***Imitator***), перетяните сюда из Панели элементов ***Станцию МЭК 60870-5-104***. В её свойствах укажите Номер станции 1 и выберите карту адресов IecTmAddressMap.

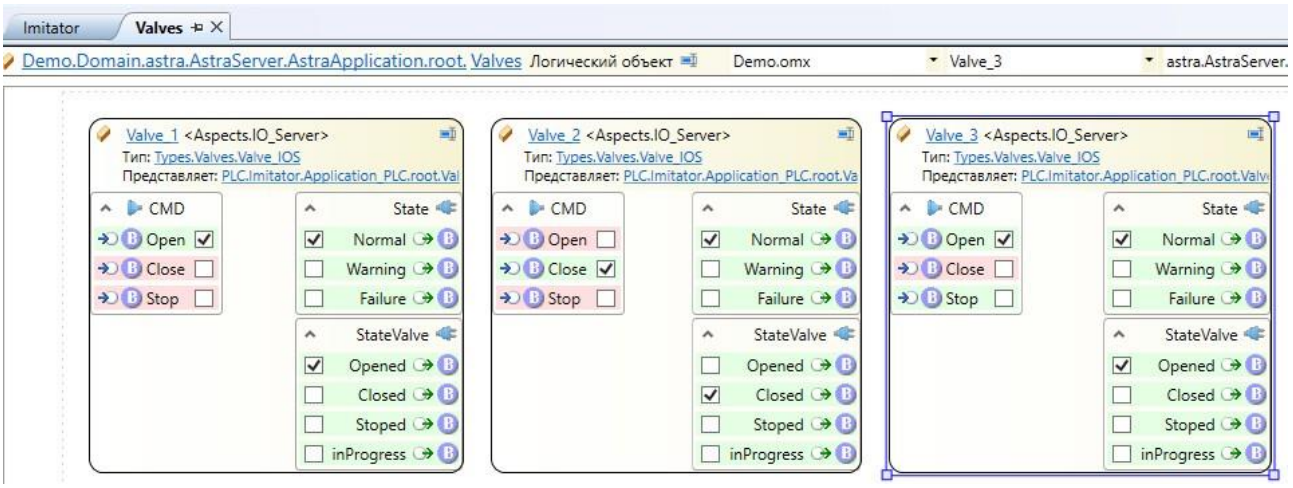


30. Перейдите в ***AstraServer*** и перетяните сюда из Панели элементов ***Опросчик Мэк 60870-5-104***.

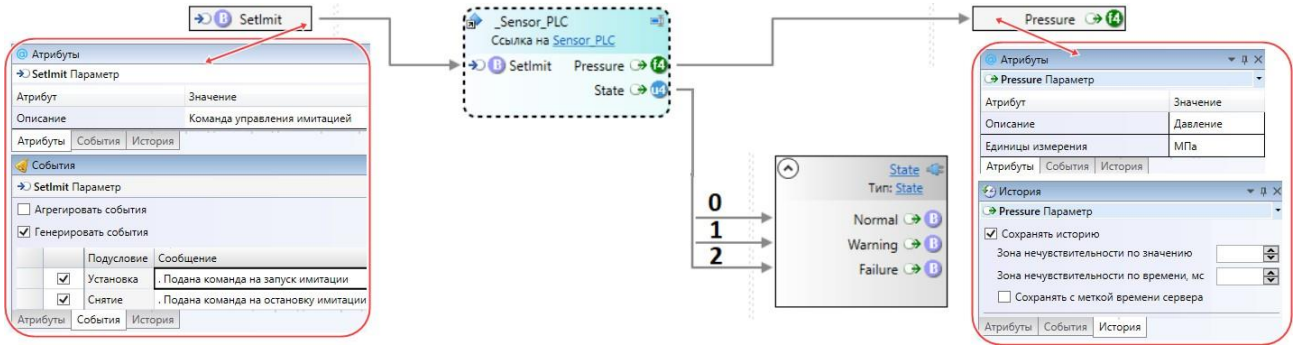


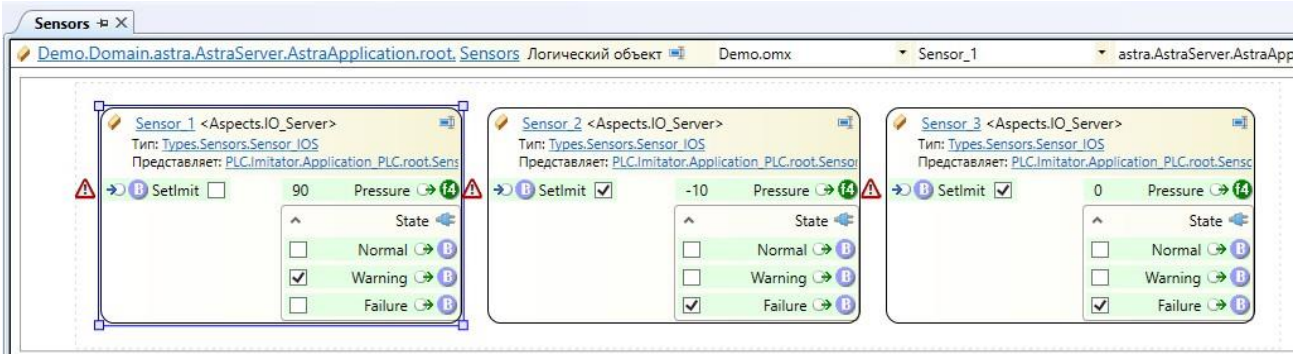
31. **Постройте решение**, **перейдите к развёртыванию** и **примените конфигурацию** к линуксовому серверу.

Для наблюдения результата перейдите в ***AstraServer*** → ***root*** → ***Valves***.



Самостоятельное задание. Создать тип датчика со стороны ПЛК и со стороны сервера ввода вывода, разместить экземпляры в папке ***root*** в ПЛК, создать представления объектов на ***AstraServer***, импортировать карту связывания (данные по датчикам передаются по Modbus, поэтому ничего нового создавать не нужно. Следует лишь дополнить уже существующую карту новыми адресами). Структура типа датчика описанного со стороны сервера ввода-вывода и конечный результат представлены на изображениях ниже.

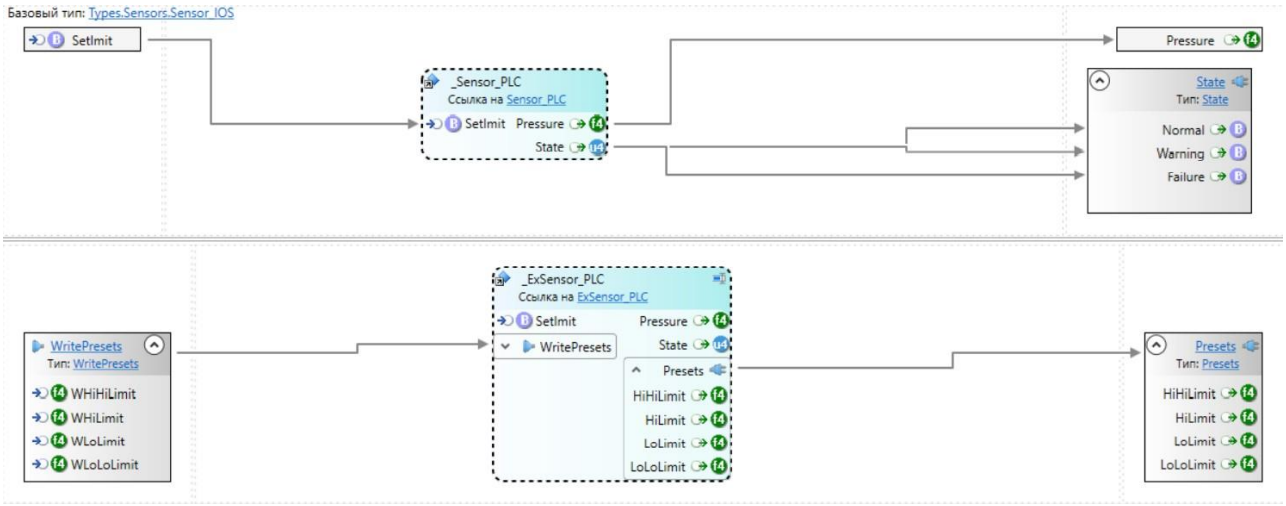




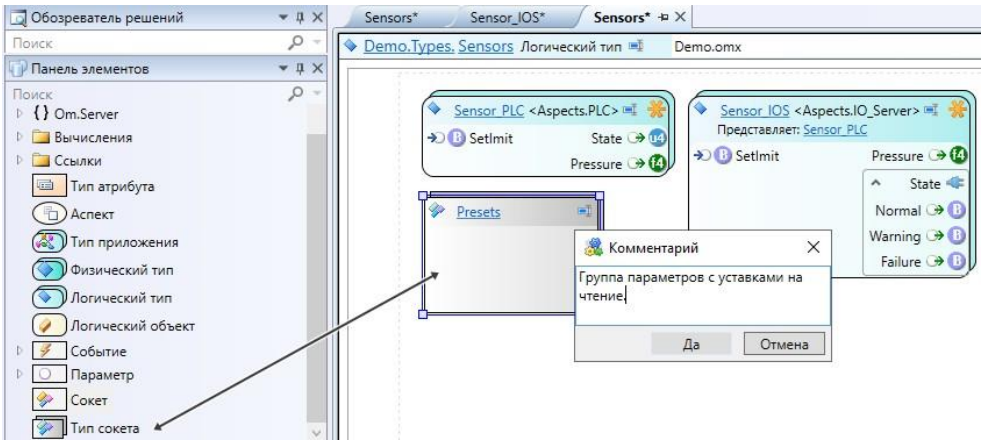
Наследование

Добавим в проект расширенные датчики и задвижки, используя наследование. Наследование позволяет описать ещё один тип, используя уже ранее описанный тип, дополнив его новыми параметрами. Для начала подготовим набор параметров, который будет использован.

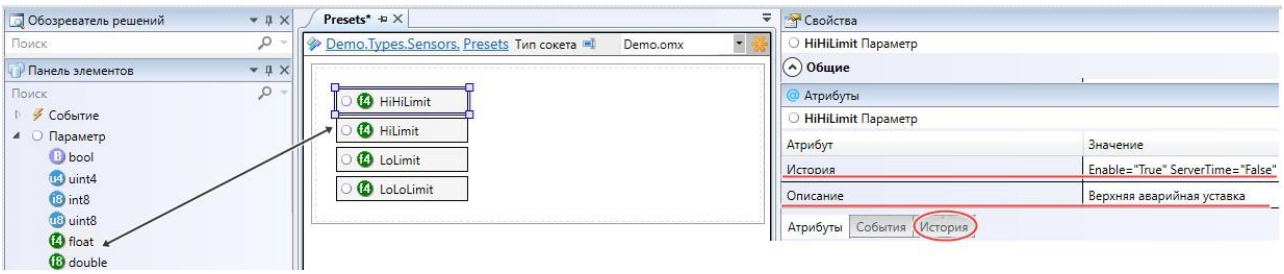
Начнём с расширенного датчика. Структура типа, описанного со стороны сервера ввода-вывода представлена ниже.



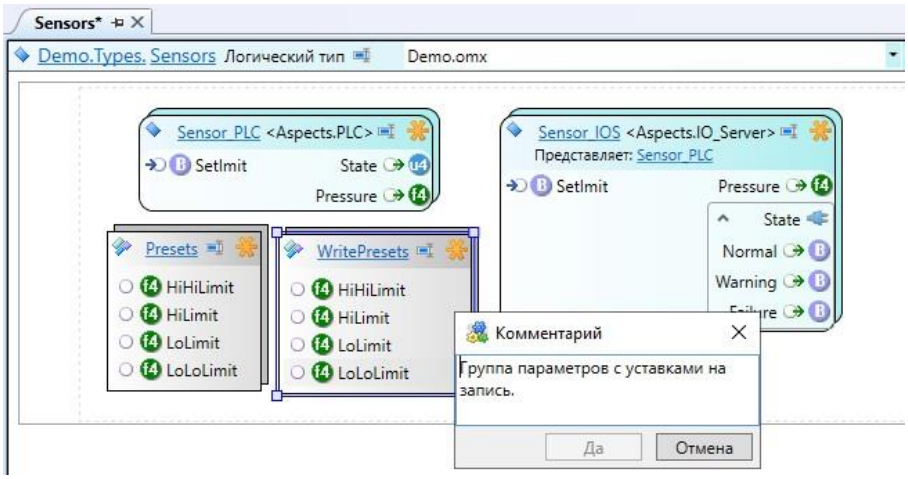
1. Перейдите в ***Пространство имён Sensors*** (Определения → Types → Sensors). Перетяните сюда из панели элементов ***Тип сокета***. Назовите его Presets и добавьте комментарий.



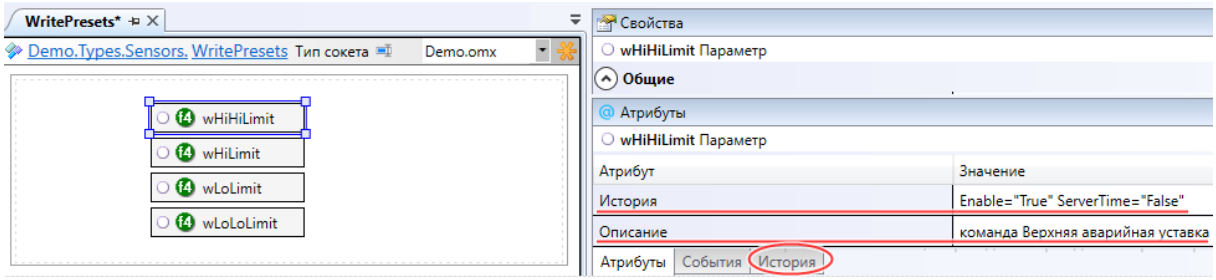
2. Перейдите внутрь ***сокета Presets*** и добавьте сюда 4 экземпляра ***параметра*** типа float. Назовите их HiHiLimit, HiLimit, LoLimit, LoLoLimit. Добавьте для всех атрибуты описания (Верхняя аварийная уставка, верхняя предаварийная уставка, нижняя предаварийная уставка, нижняя аварийная уставка) и единицы измерения (МПа). Также поставьте галочку около Сохранять историю.



3. Перейдите в ***Пространство имён Sensors***, скопируйте и вставьте ***Тип сокета Presets***. Измените название на WritePresets и комментарий.

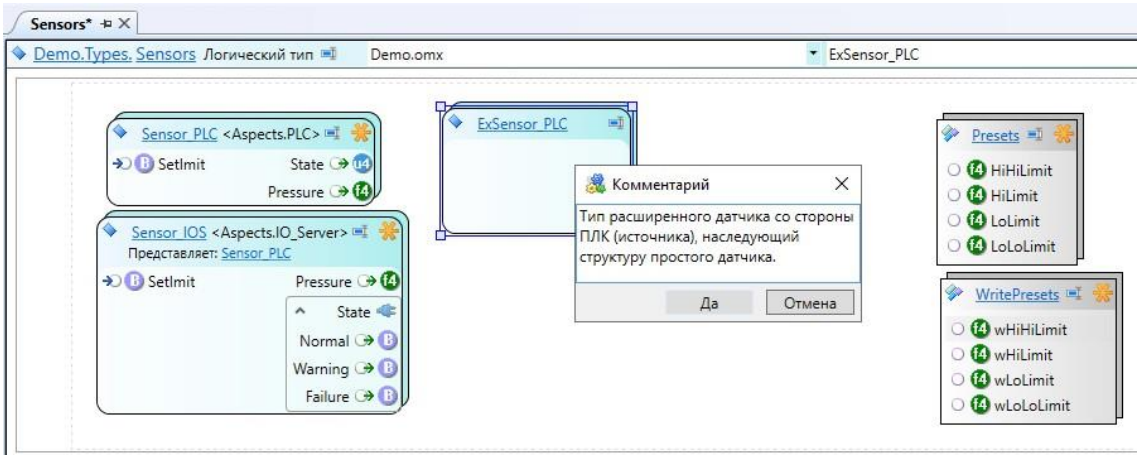


4. Перейдите внутрь ***Сокета WritePresets***. Перед имеющимся именем каждого параметра добавьте букву «w», а перед описанием – слово «команда».



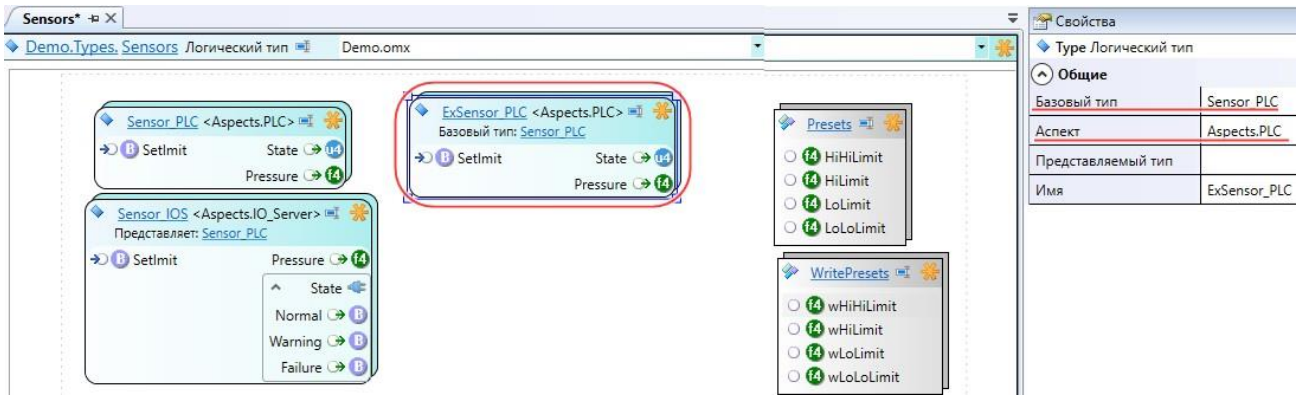
Начнём создавать логические типы.

5. Перейдите в ***Пространство имён Sensors***, перетяните сюда из Панели элементов ***Логический тип***. Назовите его ExSensor_PLC, задайте комментарий.

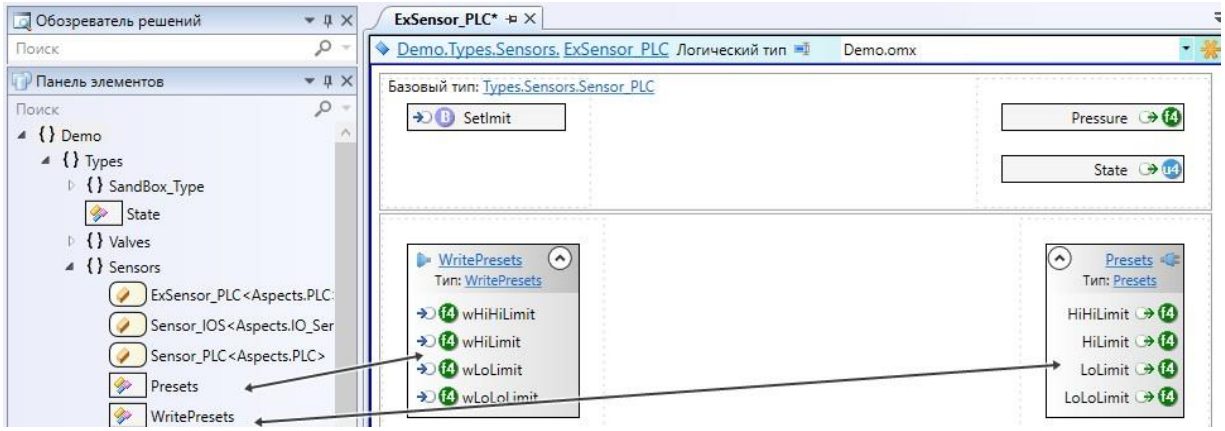


6. Выделите ***Логический тип ExSensor_PLC***, в свойстве Аспект укажите PLC, Базовый тип – Sensor_PLC.

Свойство Базовый тип отвечает за то, что именно вы будете наследовать, какую структуру и какие данные будете переносить.

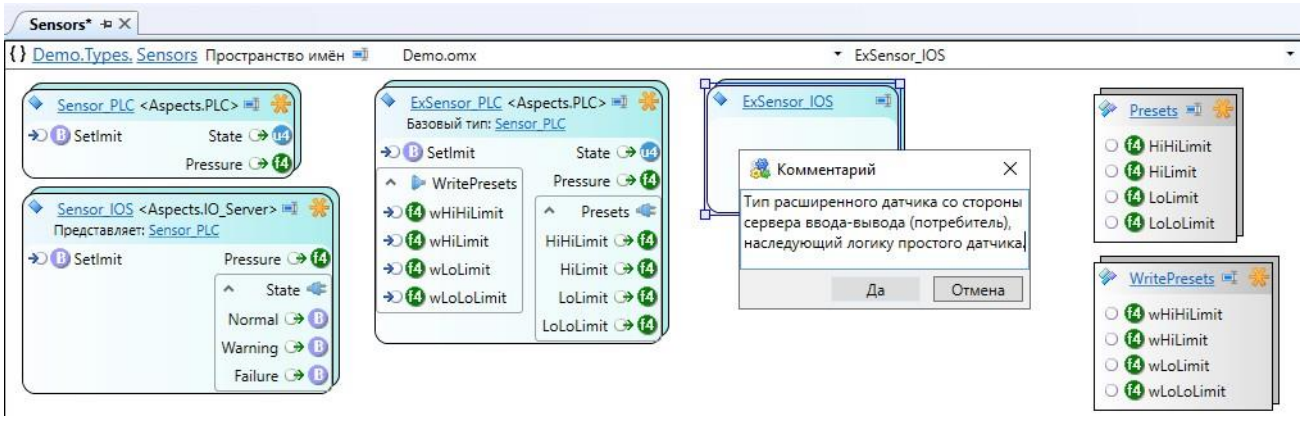


7. Перейдите в ***Логический тип ExSensor_PLC***. Перетяните сюда из Панели элементов на вход ***Тип сокета WritePresets***, а на выход – ***Presets***.

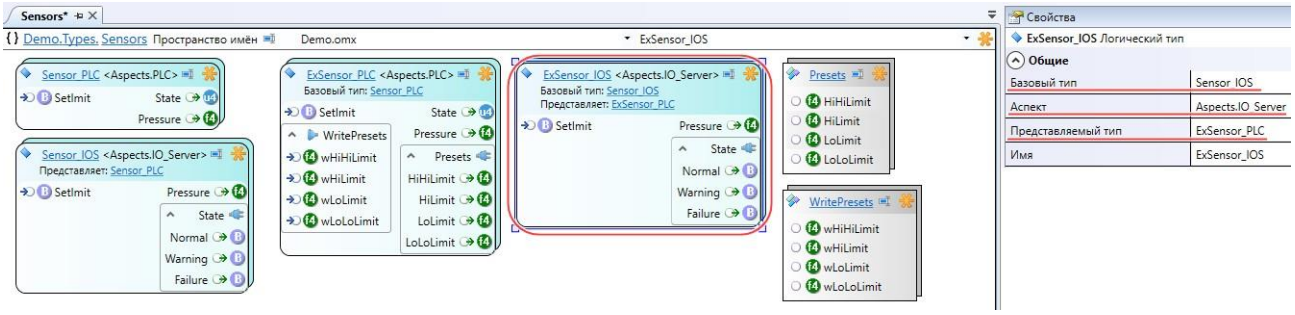


Входные и выходные параметры со стороны ПЛК описаны. Теперь необходимо описать логический тип со стороны сервера ввода-вывода.

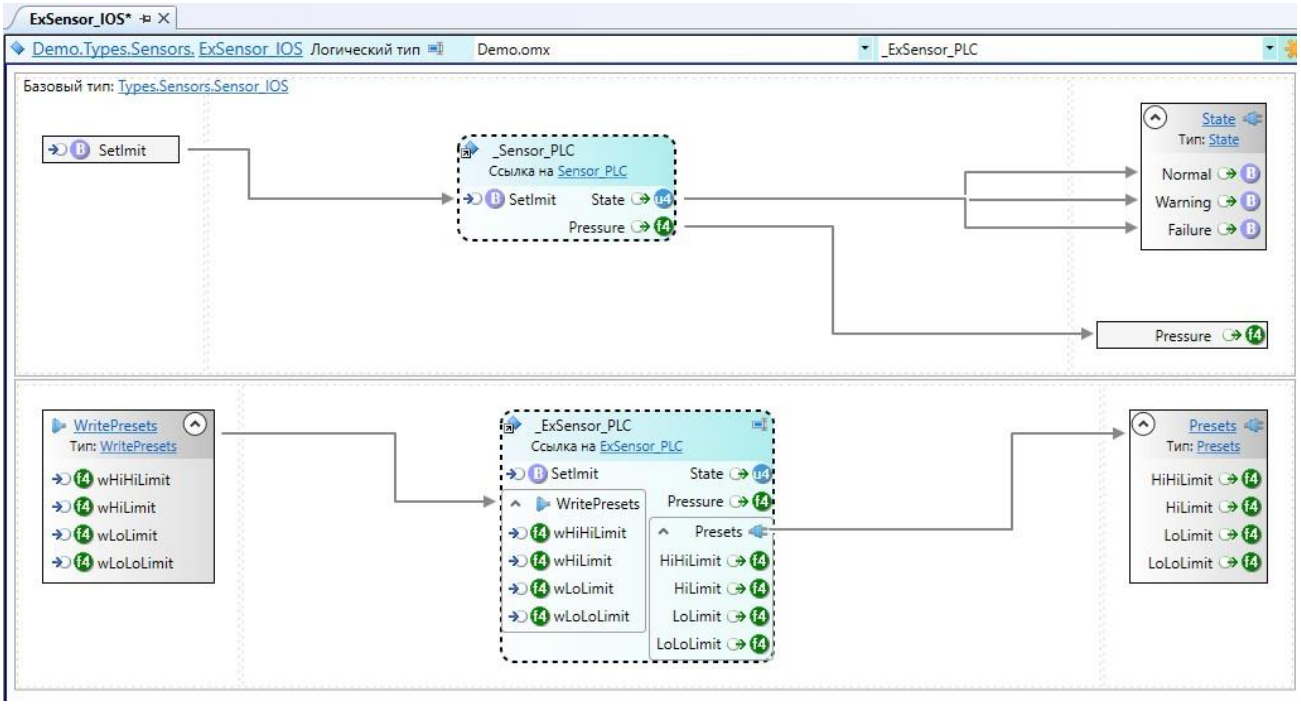
8. Перейдите в ***Пространство имён Sensors***. Перетяните сюда из Панели элементов ***Логический тип***. Назовите его ExSensor_ISO, задайте комментарий.



9. Выделите ***Логический тип ExSensor_IOS***. В свойстве Аспект укажите IO_Server, Представляемый тип – ExSensor_PLC, Базовый тип – Sensor_IOS.



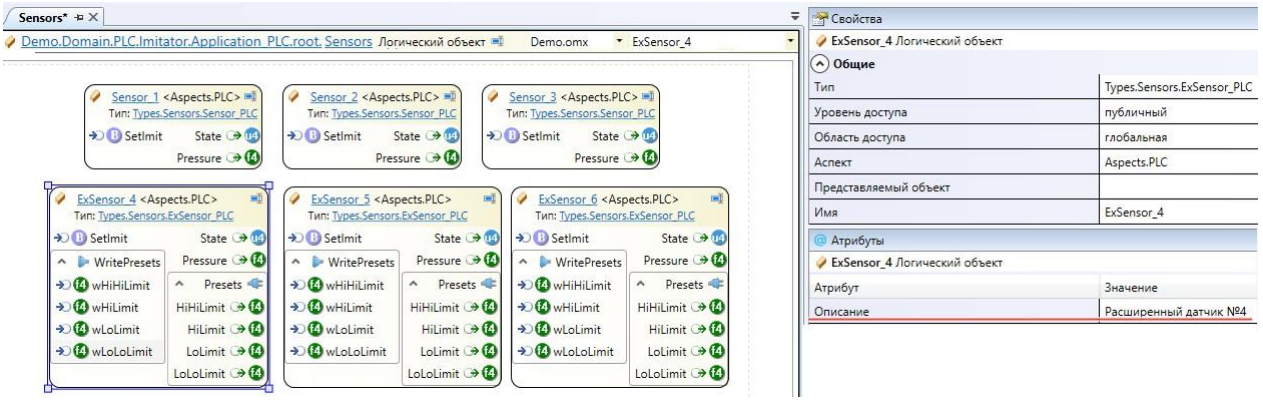
10.Перейдите внутрь ***Логического типа ExSensor_IOS***. Щелкните ПКМ по пустому рабочему полю → Добавить → Аспектную ссылку. Представление, на которое необходимо будет ссылаться – ***ExSensor_PLC***, Далее → Далее, Элементы ссылки которые требуется экспонировать – ***WritePresets*** и ***Presets***.



Так работает наследование. Логику родителя поменять нельзя, но зато можно добавить что-то новое. Но если что-то изменится у родителя, все наследники унаследуют эти изменения.

Разместим экземпляры на стороне источника и потребителя.

11.Перейдите в ***Application_PLC*** → ***root*** → ***Sensors***. Перетяните сюда из Панели элементов 3 экземпляра типа ***ExSensor_PLC***. Назовите их ExSensor_1, ExSensor_2, ExSensor_3. Каждому из них добавьте атрибут Описание (Расширенный датчик №4, 5 и 6).



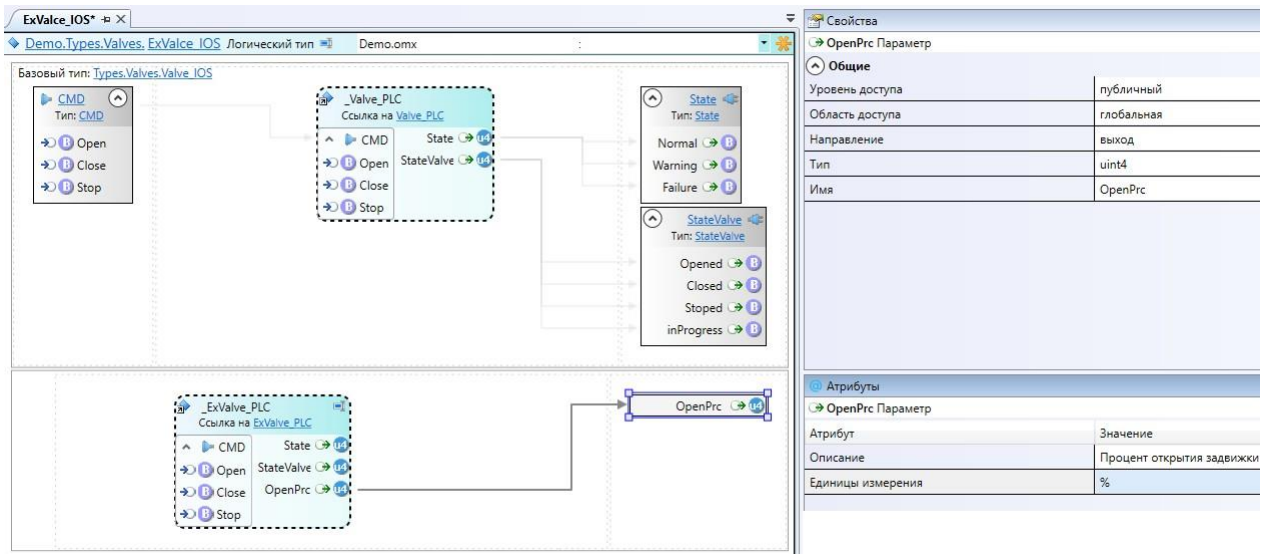
12.Перейдите в ***AstraApplication***. Щёлкните ПКМ по пустому месту на рабочем поле → Создать здесь представления объектов. Выберите всю папку ***root*** из PLC → Далее → Далее → Аспект новый представлений – Aspects.IO_Server → Готово.

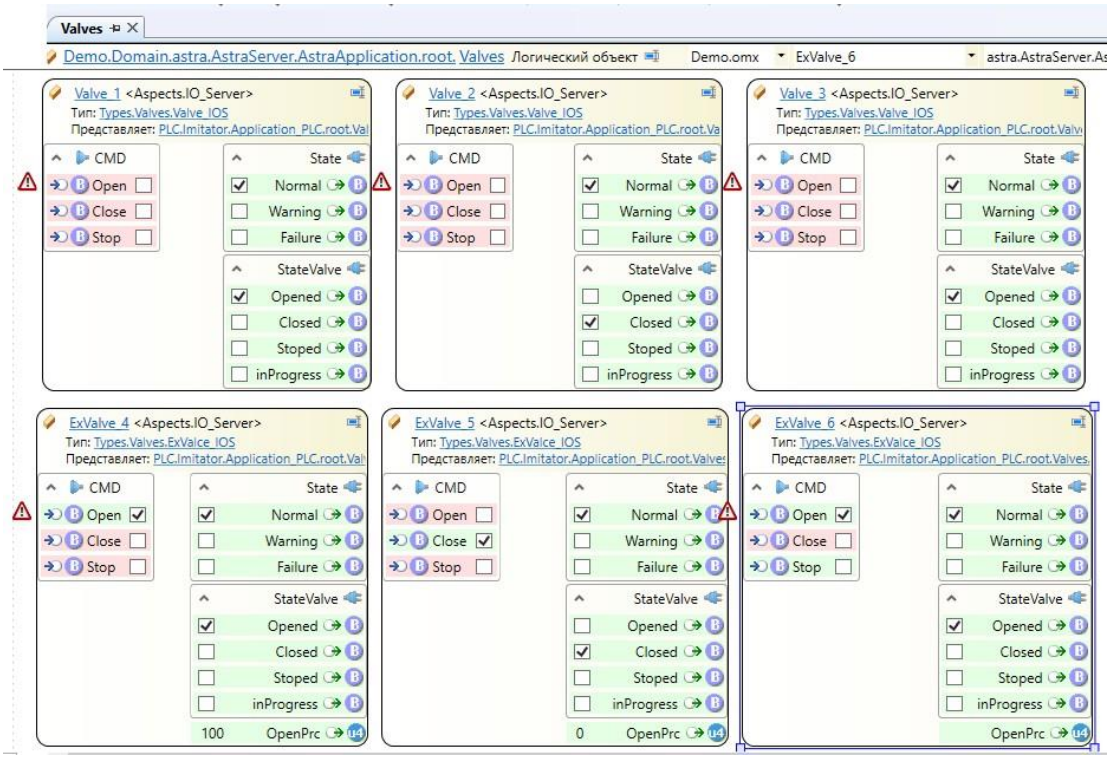
13.Перейдите в ***Application_PLC***, откройте ***Карту адресов ModbusAddressMap***, импортируйте файл mb_export_map_Sensors.xlsx. Сохраните и закройте карту адресов.

14. **Постройте решение**, **перейдите к развёртыванию** и **примените конфигурацию** к линуксовому серверу.

Для наблюдения результата перейдите в ***AstraServer*** → ***root*** → ***Sensors***.

Самостоятельное задание. Создать тип расширенной задвижки со стороны ПЛК и со стороны сервера ввода вывода, разместить экземпляры в папке ***root*** в ПЛК, создать представления объектов на ***AstraServer***, импортировать карту связывания (данные по датчикам передаются по МЭК, ничего нового создавать не нужно. Следует лишь дополнить уже существующую карту новыми адресами). Структура типа расширенной задвижки описанной со стороны сервера ввода-вывода и конечный результат представлены на изображениях ниже.

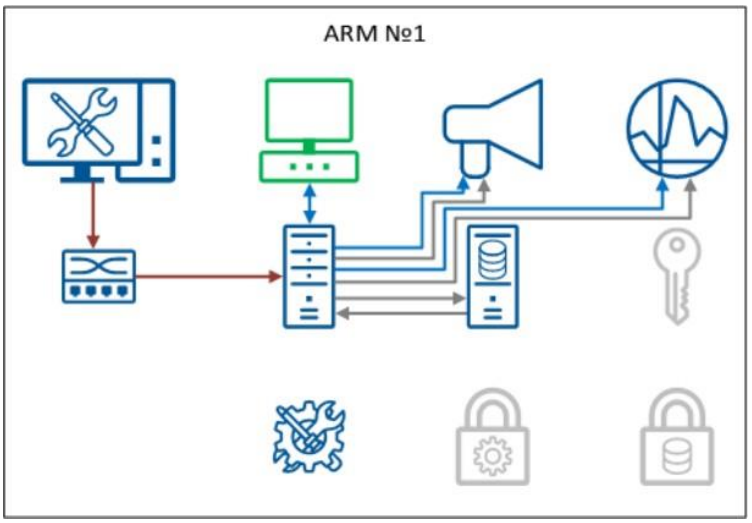




Конфигурация сервера ввода-вывода, взаимодействующим с имитатором ПЛК готова, теперь можно приступить к разработке мнемосхем.

9. SePlatform.HMI

Для того, чтобы организовать взаимодействие пользователей с технологическими данными, необходимо обеспечить графический интерфейс. Для разработки и визуализации человеко-машинного интерфейса в составе Systeme Platform платформы есть компонент *SePlatform.HMI*. В демонстрационном проекте мы визуализируем получаемые сервером данные. Перед началом работы необходимо установить *SePlatform.HMI* на машину в ОС Windows и Linux:



Установите *SePlatform.HMI* на Windows при помощи запуска дистрибутива, а на Linux через командер: перейдите в PuTTY, удостоверьтесь, что Вы находитесь в директории с дистрибутивами и после этого введите команду: **sudo dpkg –i SePlatform.hmi********.deb**.

Создание проекта

Для начала, создадим простой проект, и запустим его.

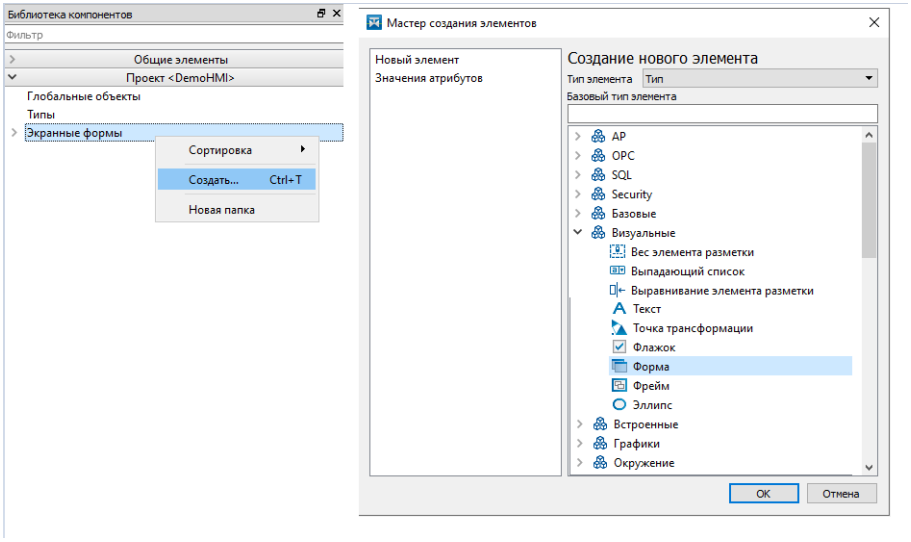
1. Запустите *SePlatform.HMI*. Пуск → SePlatform → SePlatform.HMI.
2. Нажмите Файл → Создать проект → Назовите его «DemoHMI» → Сохранить.

Проект создан, в нём уже присутствует форма по умолчанию **MainForm**. Для начала создадим пустую форму и поработаем с несколькими базовыми элементами *SePlatform.HMI*.

Добавление экранной формы

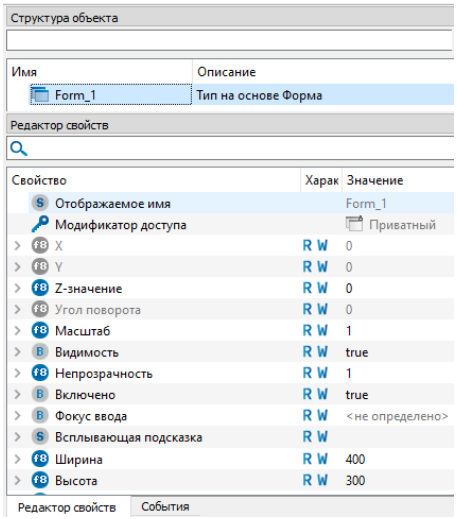
Создадим пустую экранную форму.

1. В Библиотеке компонентов раскройте список Проект → нажмите ПКМ по Экранные формы → Создать → Визуальные → Форма.



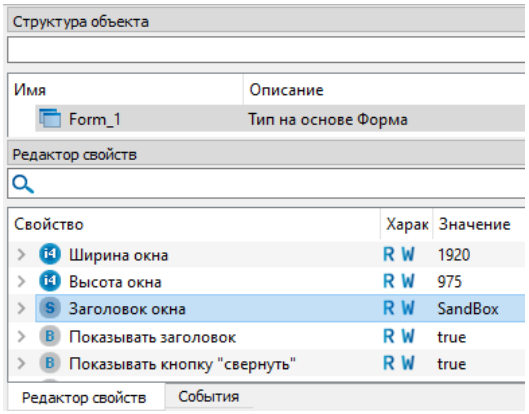
Создана пустая экранная форма. Теперь можно работать с её свойствами.

2. Откройте созданную экранную форму двойным нажатием ЛКМ.
3. Нажать ЛКМ в структуре объекта на Form_1. В редакторе свойств появятся свойства выбранного объекта.

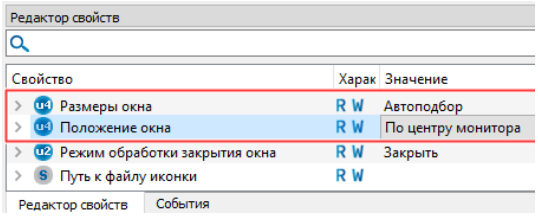


4. Для того, чтобы в RunTime форма открывалась с нужным названием, в свойстве Заголовок окна введите SandBox. Завершите ввод нажатием клавиши Enter.

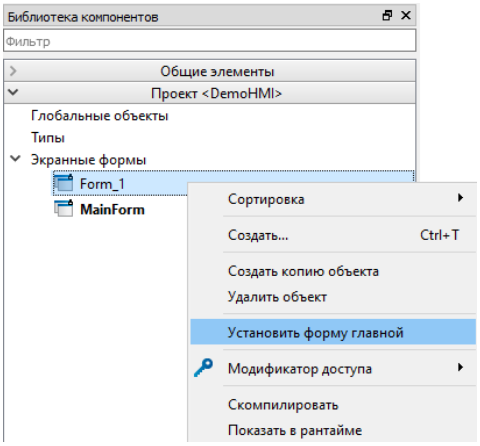
5. В свойстве Высота укажите 600.



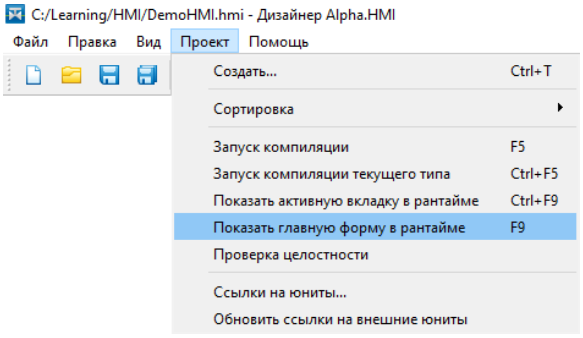
6. В свойстве Размеры окна укажите Автоподбор.
7. В свойстве Положение окна укажите По центру монитора.



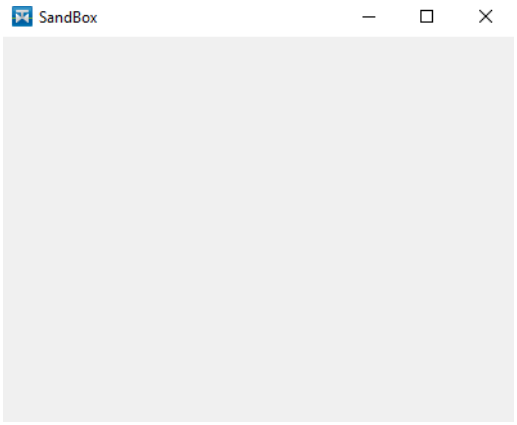
Теперь форма стандартного размера, и её можно запустить, но прежде установите форму главной: ПКМ в левой части экрана по **Form_1** → Установить форму главной.



8. Нажать Проект -> Показать главную форму в RunTime.



В результате получим пустую запущенную экранную форму. На данный момент, на экранной форме нет никакой логики и элементов.

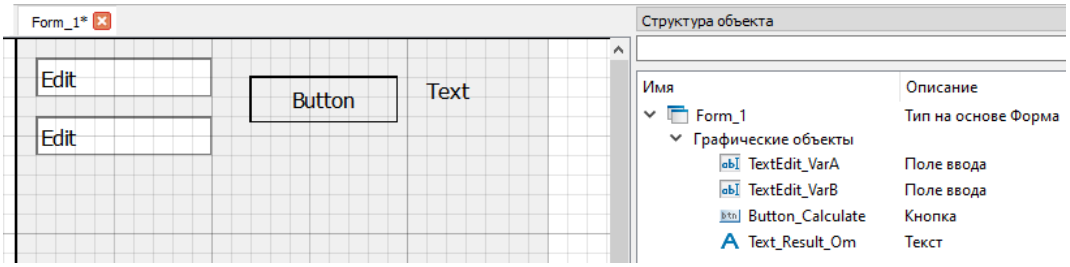


Пустая экранная форма создана, добавим на экранную форму элементы.

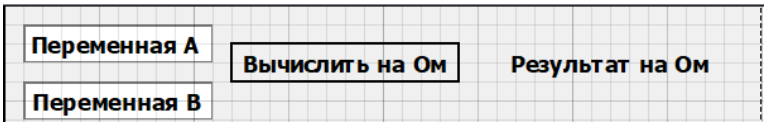
Добавление элементов

Познакомимся с некоторыми возможностями *SePlatform.HMI* на примере калькулятора. Добавим на экранную форму два поля ввода и обработчики, которые при нажатии на кнопку выведут сумму двух переменных в текстовое поле различными способами.

1. Перетяните на экранную форму из Визуальных элементов Библиотеки компонентов 2 ***поля ввода***, ***кнопку*** и элемент ***Текст***. Назовите в соответствии с изображением.

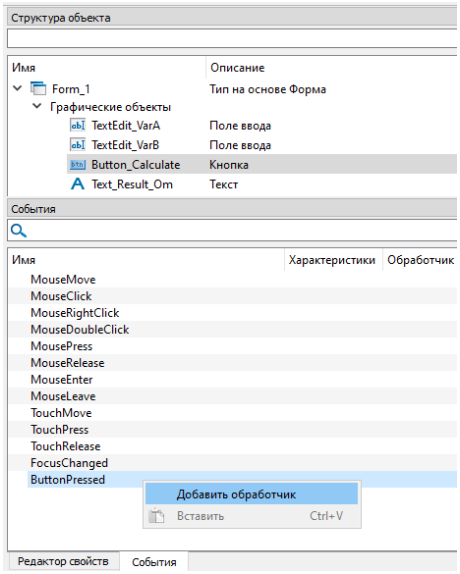


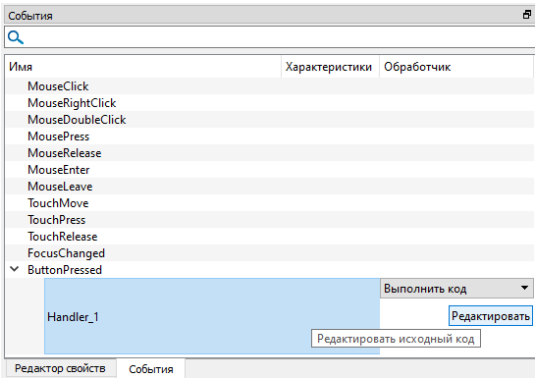
2. В свойстве Текст ***Поля ввода VarA*** укажите Переменная А (сделайте то же и для ***Поля ввода VarB)***, ***кнопки BattonCalculate*** – Вычислить на Ом, элемента ***Text_Result_Om*** – Результат на Ом. Отредактируйте шрифт и выравнивание текста.



Вычисление суммы двух переменных будет происходить по нажатию на ***Кнопку Вычислить на Ом***.

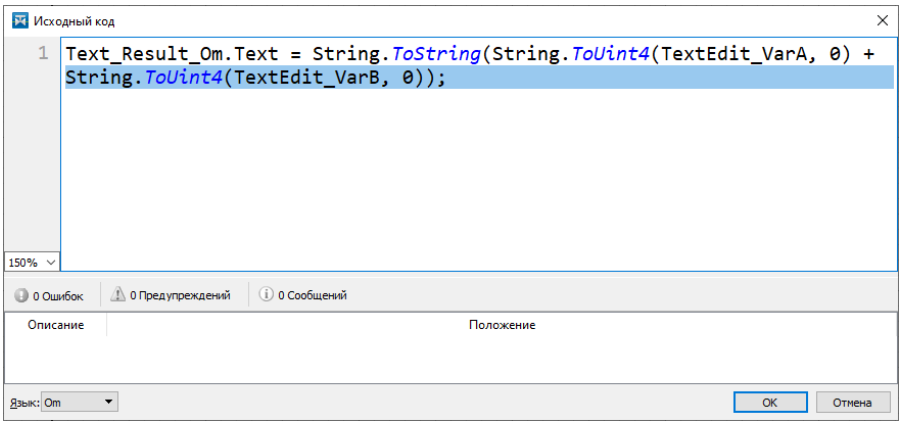
3. Выделите ***Кнопку Вычислить на Ом*** нажатием ЛКМ → перейдите во вкладку События → нажмите на событие ButtonPressed ПКМ → Добавить обработчик → Выполнить код → Редактировать.





Откроется окно, в котором можно редактировать код обработчика.

4. В коде обработчика ввести:

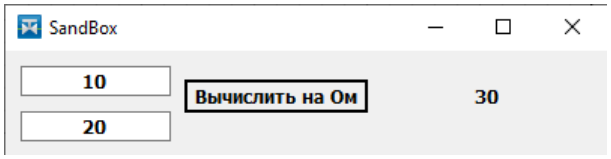


В коде обработчика можно писать скрипты как на языке SePlatform.Om, так и на JavaScript. В данном примере используется SePlatform.Om. Так как все свойства, с которыми происходит работа, имеют свой тип данных, в скрипте необходимо использовать функции, которые явно приводят строковые типы текстовых свойств к целочисленным для выполнения операции сложения. Если оставить данную операцию без преобразования типов, то в результате получится просто объединение (конкатенация) двух строк.

На языке JavaScript данный код может иметь вид: **Result.Text = +Var_A.Text + +Var_B.Text;**

5. Нажмите ОК, сохраните и запустите проект.

После запуска можно увидеть, что при нажатии на ***Кнопку Вычислить на Ом*** результат отображает сумму двух переменных.

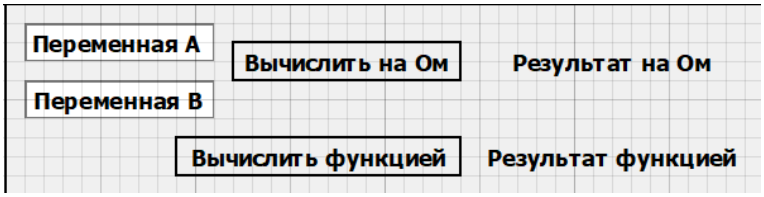


В разработке проектов предпочтительнее использовать язык SePlatform.Om так как он компилируемый (т.е. перед запуском проекта, обработчики будут проверяться на ошибки) и имеет строгую типизацию (что экономит память, улучшает быстродействие, и позволяет избежать ошибок). JavaScript
интерпретируемый (перед запуском нет проверки на ошибки, что повышает риск появления ошибок в режиме исполнения), имеет динамическую типизацию (что также способствует возникновению ошибок).

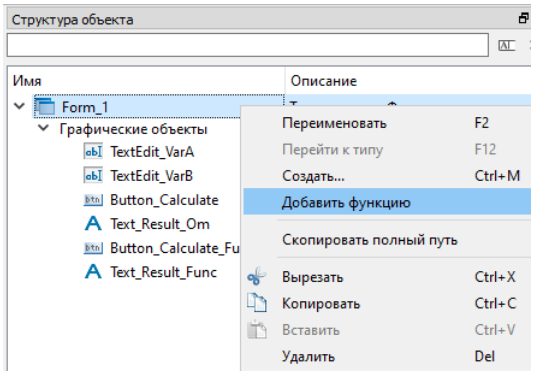
Добавление функций

Попробуем написать самописную функцию, которая будет выполнять вычисления за нас, где не нужно выполнять переводы в форматы, достаточно вызвать метод, который будет всё вычислять самостоятельно.

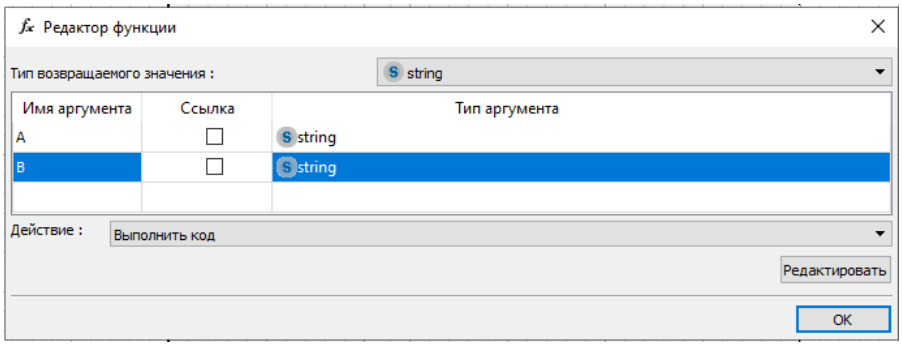
1. Перетяните на экранную форму из Библиотеки компонентов ***Кнопку*** и элемент ***Текст***. Назовите их Button_Calculate_Func и Text_Result_Func.
2. В свойстве Текст ***кнопки Button_Calculate_Func*** введите Вычислить функцией, ***Text_Result_Func*** – Результат функцией. Настройте шрифт и выравнивание текста.



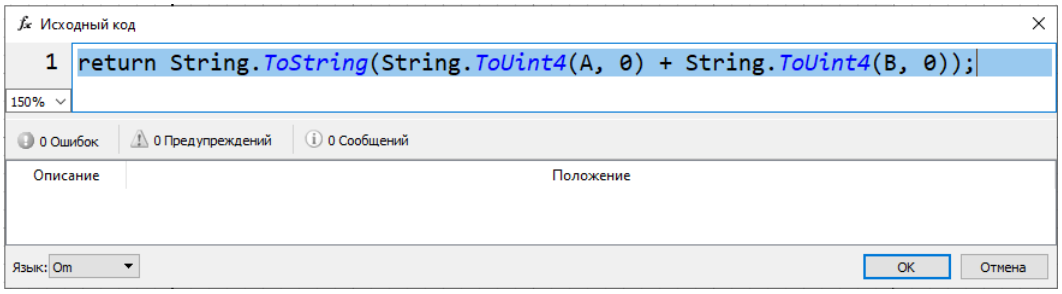
3. Для того, чтобы добавить функцию в Структуре объекта нажмите ПКМ по форме **Form_1** → Добавить функцию.



4. В Структуре объекта переименуйте функцию – Summ (функция, которая вычисляет сумму).
5. В Структуре объекта щелкните дважды по функции. Откроется редактор функции.
6. В редакторе функции добавьте 2 аргумента A и B, установите им **Тип** аргумента string. В поле Действие выберите **Выполнить код**. В поле **Тип** возвращаемого значения выберите string. Нажмите Редактировать.



7. В открывшемся окне введите код:

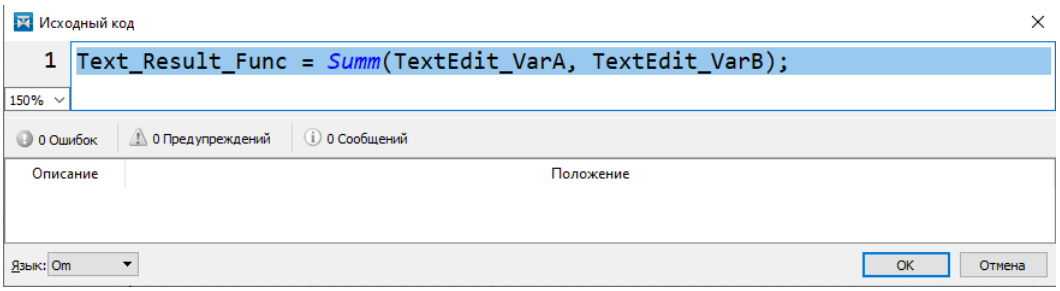


8. Нажмите ОК.

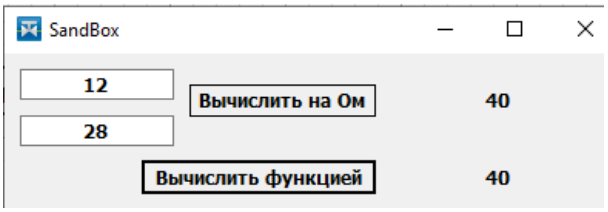
Теперь нужно где-то вызвать эту функцию. Функция будет вызвана при нажатии на ***Кнопку Вычислить функцией***.

9. Выделите ***Кнопку Вычислить функцией***, перейдите к её событиям. ПКМ нажмите на ButtonPressed → Выполнить код → Редактировать.

10. В открывшемся окне введите код:



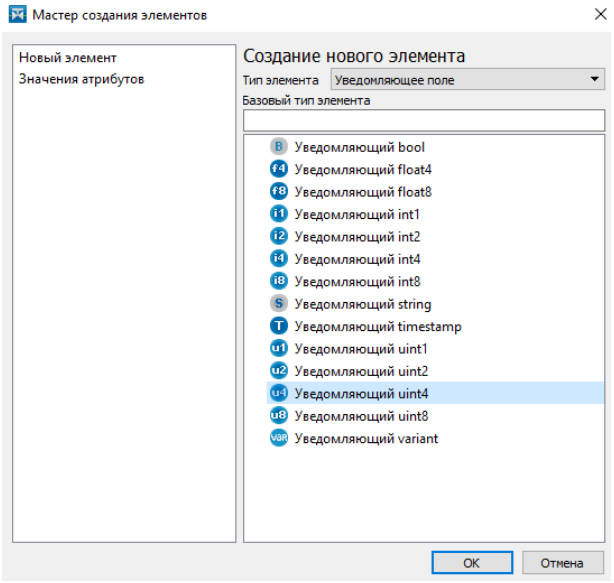
11. Нажмите ОК, сохраните и запустите проект. Теперь вычисления производятся и при помощи созданной функции.



Добавление внешних переменных

Чтобы каждое изменение свойства объекта приводило к возникновению события, применяются уведомляющие поля. Уведомляющие поля удобны, когда требуется оперативная реакция на любое изменение важного параметра.

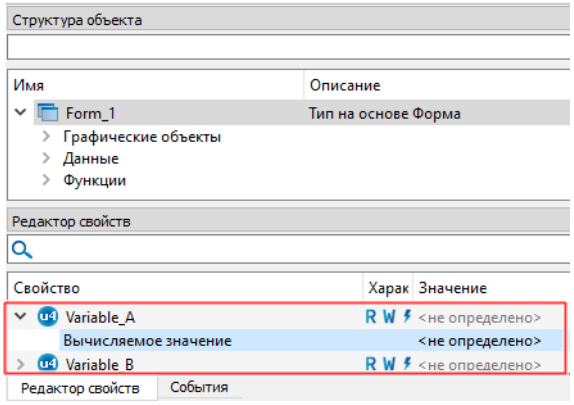
1. Для того, чтобы создать ***Уведомляющее поле***, нажмите ПКМ на **Form_1** в структуре объекта → Создать. В поле Тип элемента выберите Уведомляющее поле. В поле Базовый тип элемента выберите Уведомляющий uint4. Нажмите ОК.



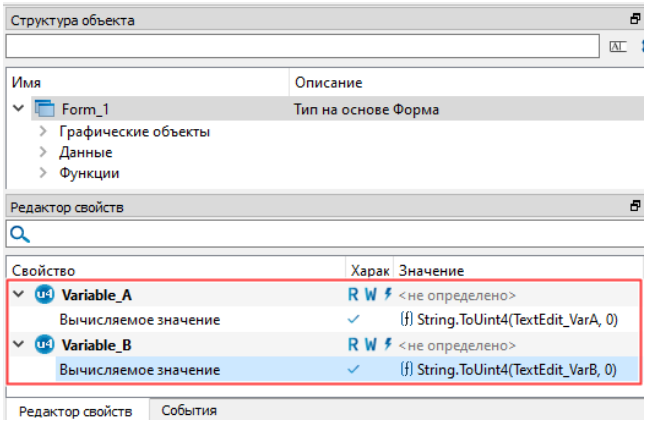
Уведомляющее поле выбранного типа добавится в Структуру объекта (группа Данные) и в список свойств объекта.

2. В Структуре объекта назовите это ***Уведомляющее поле*** Variable_A.
3. Создайте ещё одно ***Уведомляющее поле*** с именем Variable_B.

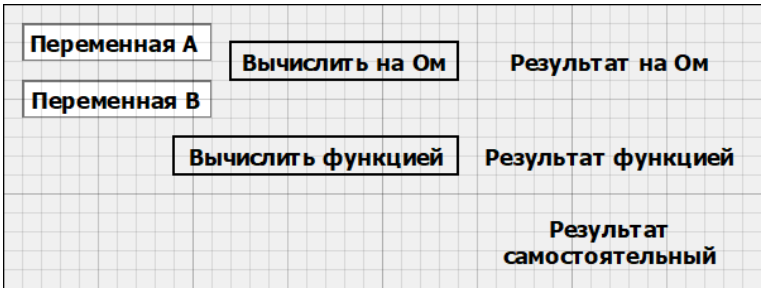
Эти ***Уведомляющее поля*** были добавлены через форму **Form_1**, значит в свойствах **Form_1** появились новые пункты: ***Variable_A*** и ***Variable_B***. И в их Вычисляемом значении можно указать, что именно поместить в это свойство.



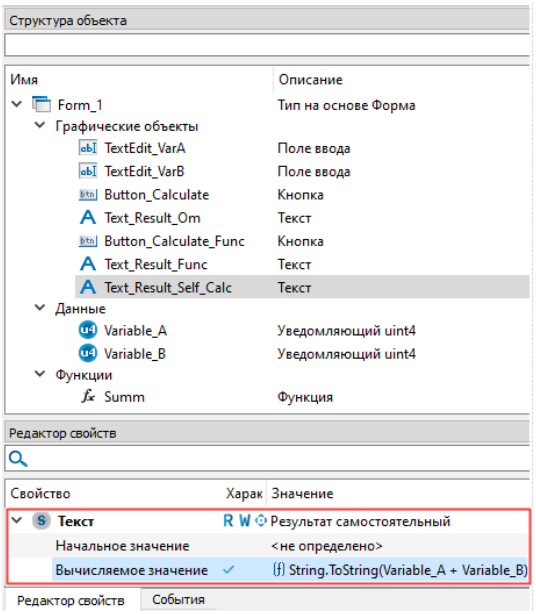
4. Поместим в Вычисляемое значение этих ***Уведомляющих полей*** код для того, чтобы эти 2 переменные вычислялись автоматически:



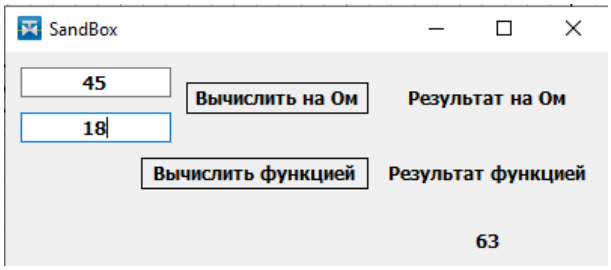
5. Добавьте на экранную форму из Библиотеки компонентов Визуальный элемент ***Текст***, назовите его в Структуре объекта Text_Result_Self_Calc, в свойстве Текст введите Результат самостоятельный. Настройте шрифт и выравнивание текста.



6. Выделите элемент ***Text_Result_Self_Calc***, перейдите к его вычисляемому значению свойства Текст и введите код:



7. Сохраните проект, запустите. Теперь в ***Текстовом поле Результат самостоятельный*** вычисление производится автоматически.



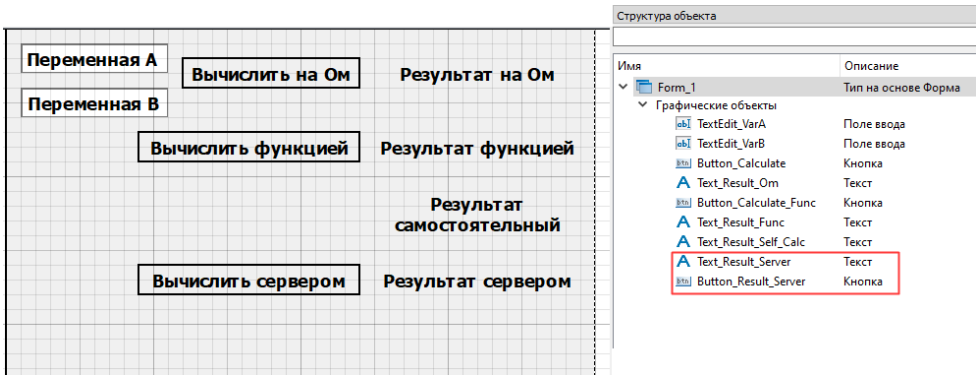
Работа с элементами AP

Элементы AP – набор компонентов для взаимодействия с сигналами определенного типа по протоколу TCP. Компонент не визуальный (не отображается на форме) и виден только в области Структуры объекта.

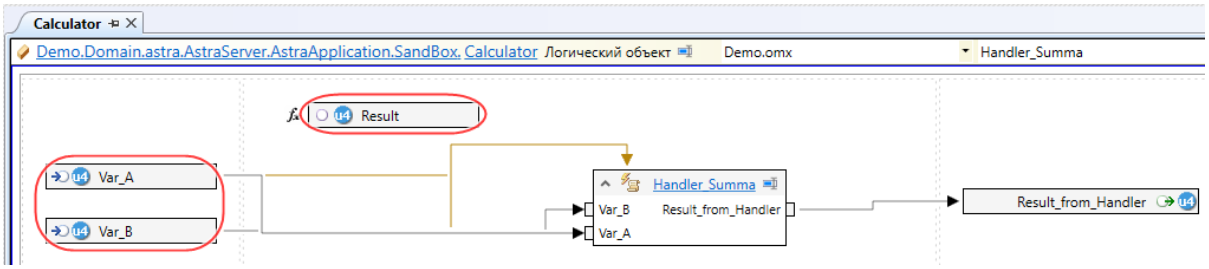
Добавление элементов AP определённого вида

В проекте *DevStudio* внутри ***SandBox*** есть ***Calculator***, где он также может выполнять вычисления. Значит, можно сделать так, чтобы вычисления производил не *SePlatform.HMI*, а *SePlatform.Server*, то есть будем записывать данные в сервер, а потом эти данные с сервера считывать как раз через *SePlatform.HMI*.

1. Добавьте на экранную форму **Form_1** ещё одну ***Кнопку*** (с именем Button_Calculate_Server и надписью внутри Вычислить сервером) и элемент ***Текст*** (с именем Text_Result_Server и надписью внутри Результат сервером). Настройте шрифт и выравнивание текста.



Для работы с сервером следует использовать ***Элементы AP***. Для того, чтобы подключиться к серверу к определенному сигналу, нужно добавить ***Элемент AP*** соответствующего типа. В проекте в *DevStudio* параметры ***Var_A***, ***Var_B*** и ***Result*** внутри ***Calculator*** типа Uint4, соответственно и в *SePlatform.HMI* нужно добавлять ***Элементы AP*** типа uint4 из библиотеки компонентов.

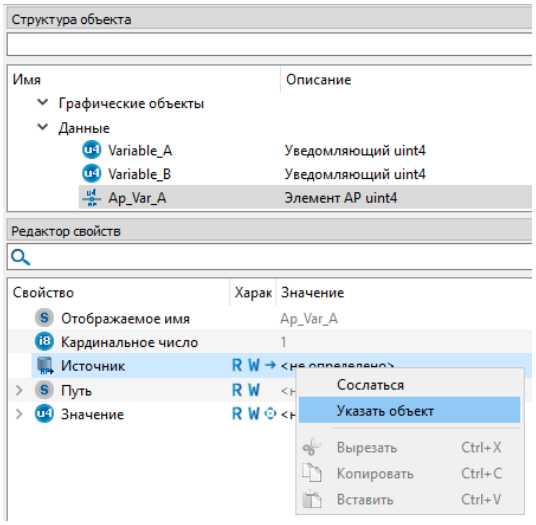


2. Перетяните на экранную форму **Form_1** ***Элемент AP*** uint4 из Библиотеки компонентов (раздел AP). В структуре объекта назовите его Ap_Var_A.

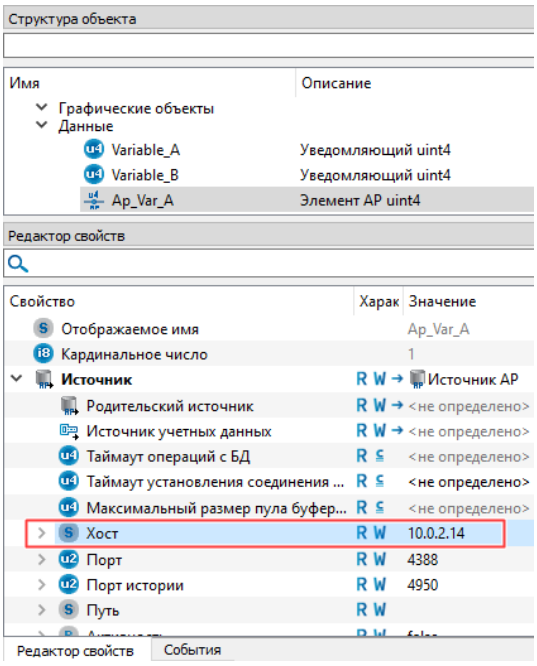
Это не визуальный компонент, отображается только в данных Структуры объекта.

Теперь этому ***Элементу AP*** нужно показать, как правильно подключаться к серверу. То есть ему нужно указать IP-адрес машины, на которой крутится сервер (в нашем случае – машина с ОС Linux) и то, по какому пути ему нужно будет подключиться.

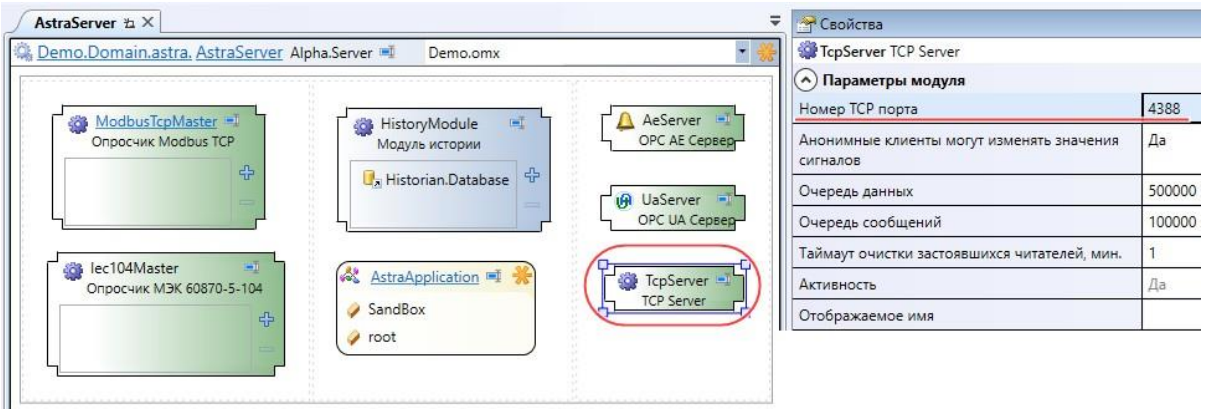
3. Выделите в Структуре объекта ***Элемент Ap_Var_A*** и щелкните ПКМ по его свойству Источник → Указать объект.

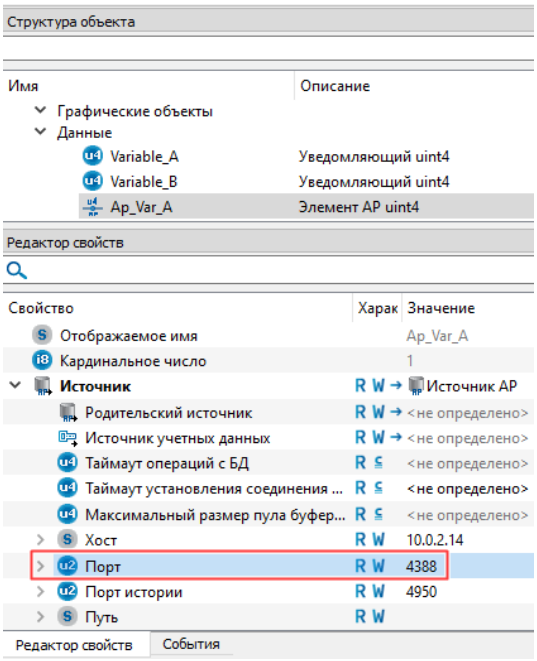


4. В свойстве Источник есть пункт Хост. Здесь необходимо указать IP-адрес машины, на которой крутится сервер.



5. В свойстве Источник есть пункт Порт. Здесь должен быть тот же самый номер порта, что и у ***Модуля TCP Server*** внутри ***AsrtaServer*** в проекте ***DevStudio***.

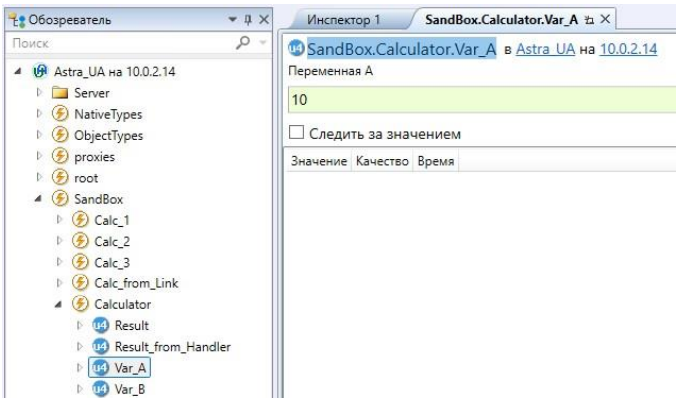


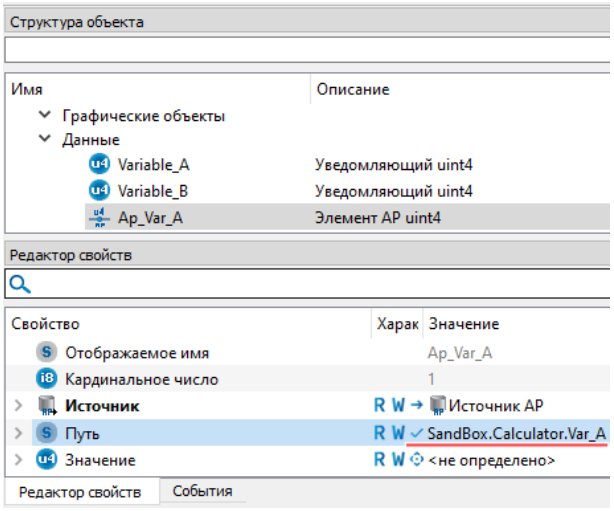


6. В свойстве Активность укажите True.
7. Сверните свойство Источник.

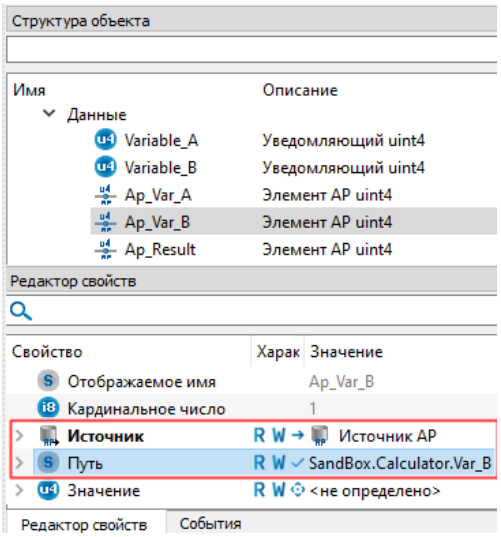
В свойстве Путь необходимо указать путь, по которому этот элемент сможет достучаться до конкретного сигнала в сервере. Тег сигнала можно найти в *OpcExplorer*.

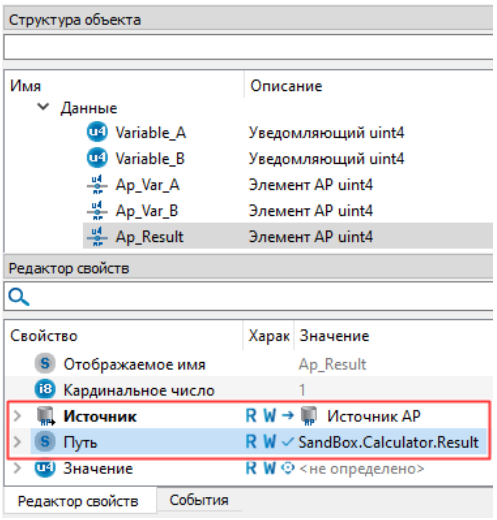
8. Откройте *OpcExplorer*, в Обозревателе найдите ***Параметр Var_A***, щелкните ЛКМ. Откроется окно, в котором прописан путь до этого сигнала, скопируйте его и вставьте в свойство Путь элемента ***Ap_Var_A***.





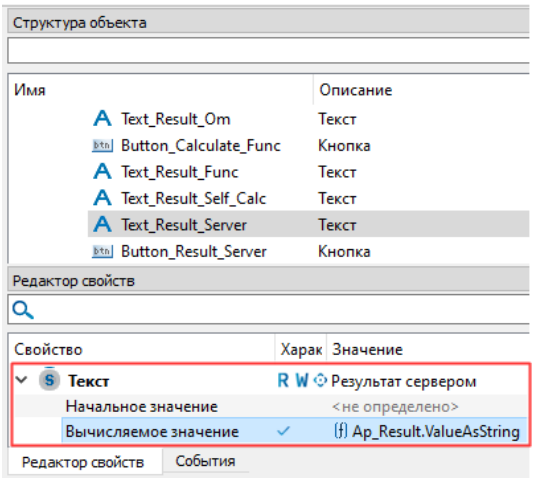
9. Проделайте действия 2-8 для ***Var_B*** и ***Result***.





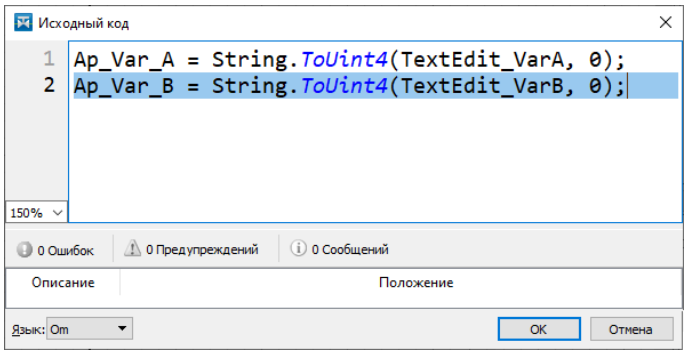
Результат вычисления будем сразу выводить в ***текстовое поле Результат сервером***.

10.Выделите в Структуре объекта ***Text_Result_Server***, перейдите к его Вычисляемому значению свойства Текст. Введите код:

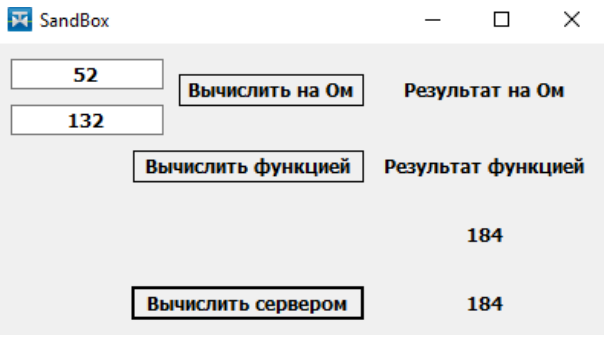


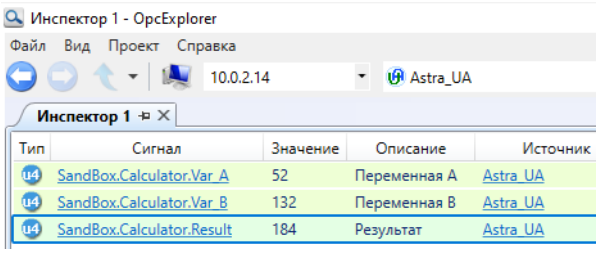
Значение в сервер мы будем записывать при нажатии на ***Кнопку Вычислить сервером***.

11.В Структуре объекта выделите ***Кнопку Button_Calculate_Server***, перейдите во вкладку События, щелкните ПКМ по ButtonPressed → Добавить обработчик → Выполнить код → Редактировать. Введите код:



12.Нажмите ОК, сохраните проект, запустите. Проверьте соответствие значений сигналов ***Var_A***, ***Var_B*** и ***Result*** в Инспекторе *OpcExplorer*.





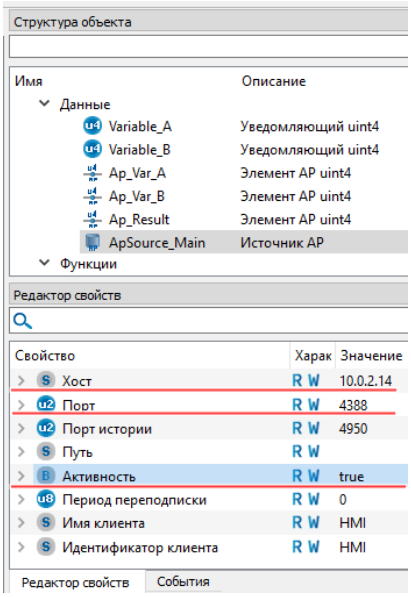
Добавление источников данных

Источник АР – компонент для связи с источником данных по протоколу TCP. Компонент невизуальный (не отображается на форме) и виден только в области Структура объекта.

В случае, если придется изменить параметры подписки на сигналы, например, порт или хост, то придется вносить изменения в каждый ***Элемент АР***. Это неудобно, для унификации есть возможность создать ***Источник АР***, где будут указаны все необходимые свойства для подключения.

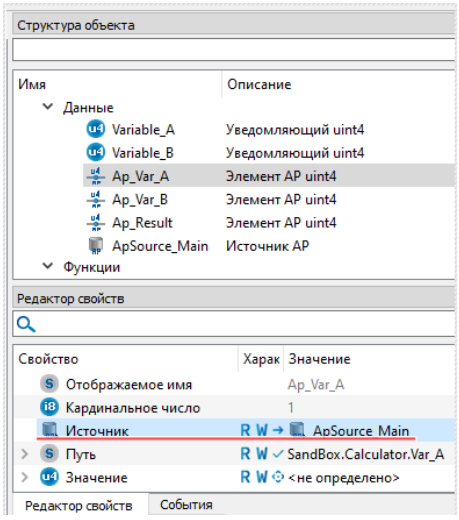
1. Перетяните на экранную форму из Библиотеки компонентов ***Источник АР*** (раздел АР). В Структуре объекта переименуйте его в ApSource_Main.

2. Выделите ***ApSource_Main*** в Структуре объект, в свойстве Хост введите IP-адрес машина, на которой находится сервер, в свойстве Порт должен быть тот же самый номер порта, что и у ***Модуля TCP Server*** внутри ***AsrtaServer*** в проекте ***DevStudio***, Активность – true.



Теперь элементы ***Ap_Var_A***, ***Ap_Var_B*** и ***Ap_Result*** должны получать данные о хосте и порте от этого источника.

3. В Структуре объекта выделите ***Ap_Var_A***, нажмите ПКМ по свойству Источник → Сброс значения.
4. В структуре объекта выделите ***Ap_Var_A***, нажмите ПКМ по свойству Источник → Сослаться → ***ApSource_Main***.

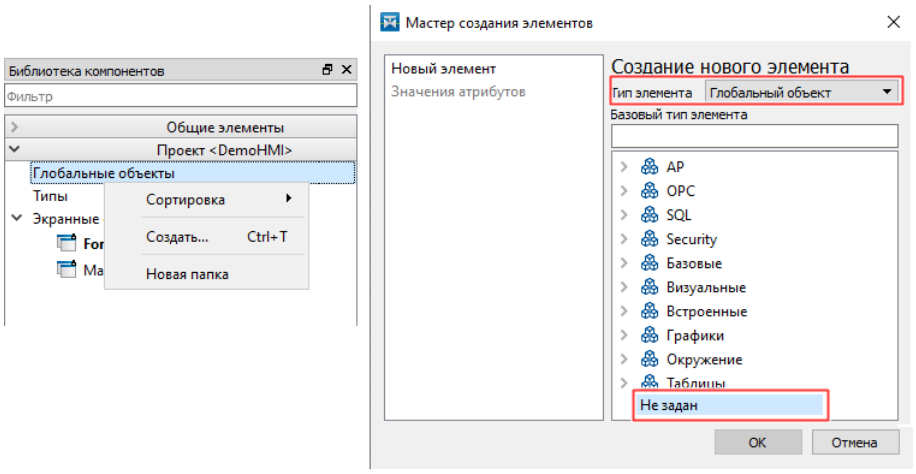


5. Повторите действия 3, 4 для ***Ap_Var_B*** и ***Ap_Result***.
6. Сохраните проект, запустите. Проверьте соответствие значений сигналов ***Var_A***, ***Var_B*** и ***Result*** в Инспекторе *OpcExplorer*.

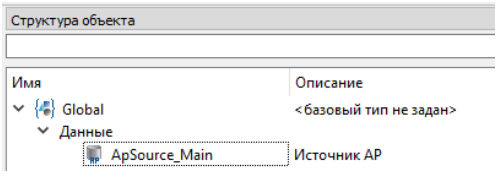
Использование глобальных объектов. Каскадирование источников

В проекте может быть использовано много экранных форм и различных объектов, использующих общие ресурсы. Например, подключение к OPC серверу. Для того, чтобы на каждую экранную форму не добавлять Источники AP, где прописан полный путь, и в случае, если необходимо внести изменения, не редактировать их в каждой форме по-отдельности, можно использовать Глобальные объекты.

1. В библиотеке компонентов во вкладке Проект щёлкните ПКМ на Глобальные объекты → Создать. Откроется Мастер создания элементов. В поле Тип элемента выберите Глобальный объект, Базовый тип элемента - Не задан. Назовите его Global.



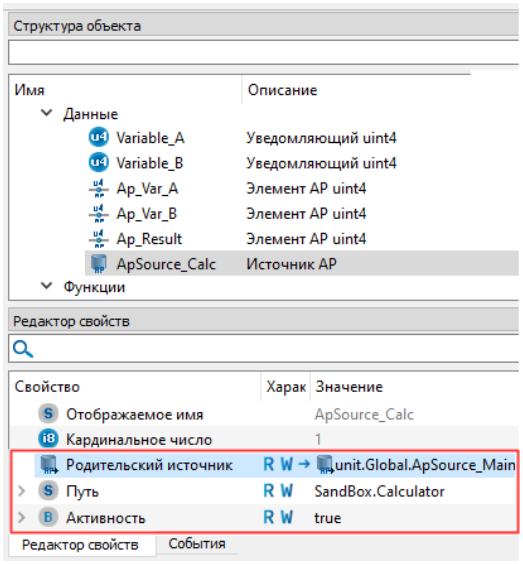
2. Откройте глобальный объект **Global** двойным нажатием мыши.
3. Перейдите в форму **Form_1**, вырежьте отсюда источник **ApSource_Main** и вставьте его в **Global**.



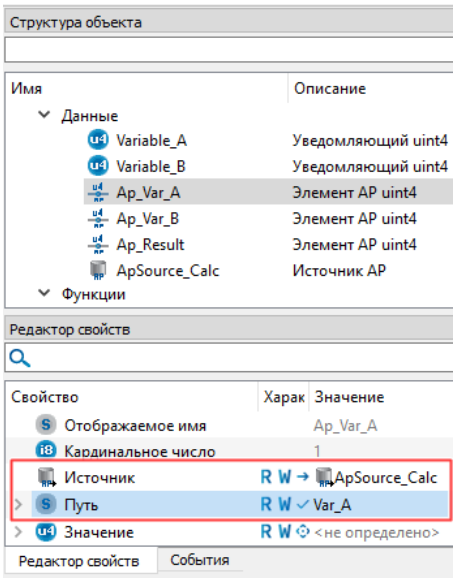
Теперь на этот источник можно ссылаться из любого места в проекте при помощи специального слова **unit**.

Так как у элементов ***Ap_Var_A***, ***Ap_Var_B*** и ***Ap_Result*** есть одинаковая часть пути (SandBox.Calculator), можно эту часть вложить в ещё один ***Источник AP***.

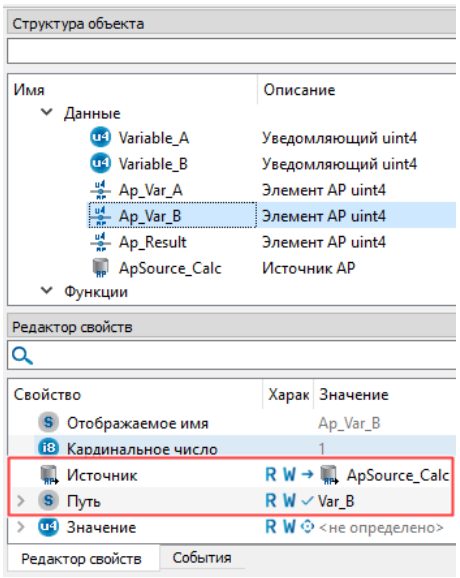
4. Перейдите в форму **Form_1**, перетяните из Библиотеки компонентов ***Источник AP***, назовите его ApSource_Calc.
5. Выделите в Структуре объекта ***ApSource_Calc***, щёлкните ПКМ по свойству Родительский источник → Сослаться → **unit.Global.ApSource_Main**. В свойстве Активность укажите true, а в свойстве Путь укажите одинаковую часть пути: **SandBox.Calculator**.

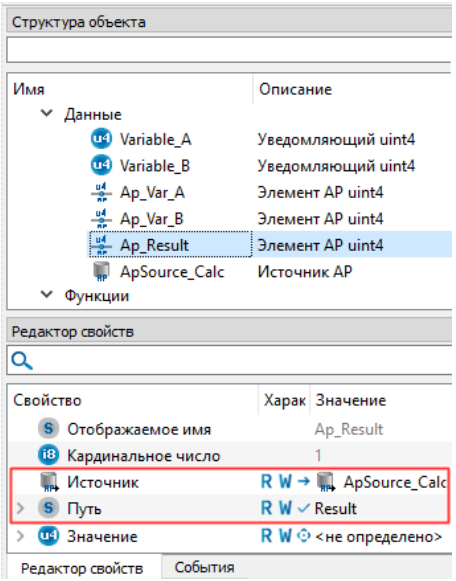


6. В структуре объекта выделите элемент ***Ap_Var_A***, щёлкните ПКМ по свойству Источник → Сослаться → ApSource_Calc. В свойстве Путь оставьте только индивидуальную часть пути: **Var_A** (то есть Вы каскадируете источники, собираете путь по кусочкам).



7. Проделайте действия из пункта 6 с элементами ***Ap_Var_B*** и ***Ap_Result***.





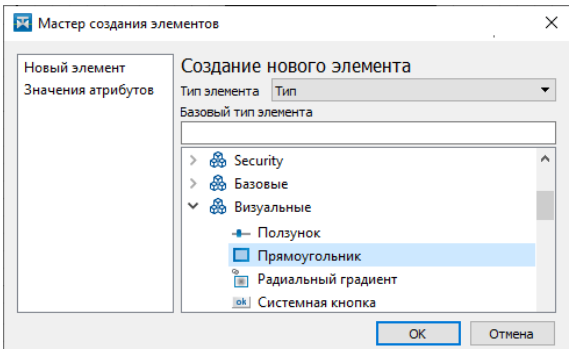
8. Сохраните проект, запустите. Проверьте соответствие значений сигналов ***Var_A***, ***Var_B*** и ***Result*** в Инспекторе *OpcExplorer*.

В результате сервер также будет производить вычисления. Но для элементов OPC, которые используются в проекте, используется источник, указанный в глобальных объектах. Таким образом, можно облегчить работу с большим количеством экранных форм и объектов в проекте. При внесении изменений не будет необходимости редактировать каждый источник, достаточно поменять свойства в глобальном объекте.

Типизация

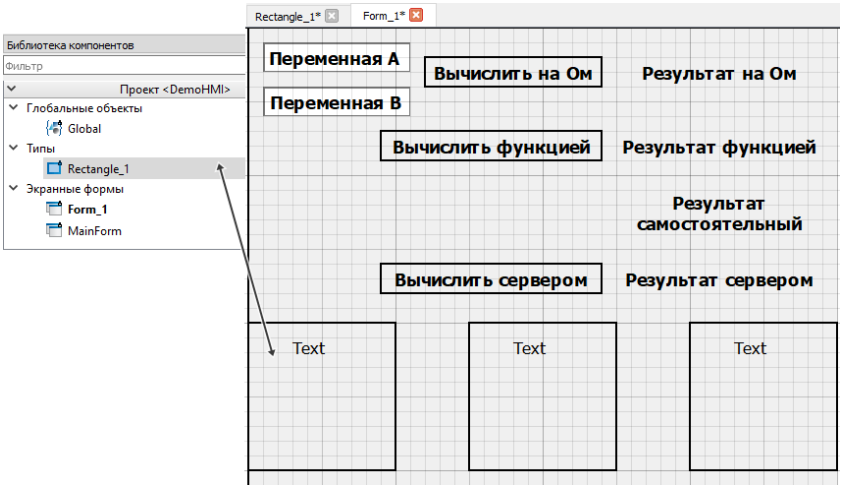
В проекте может быть много типовых элементов, например датчиков, которые могут быть использованы на различных экранных формах. Для того, чтобы постоянно не рисовать датчик на мнемосхеме и не добавлять логику в каждый из них, можно один раз создать типовой датчик, затем тиражировать его экземпляры на мнемосхемах. Если необходимо будет внести изменения, то нужно будет всего лишь отредактировать тип датчика, и все экземпляры унаследуют эти изменениями. Это позволяет избежать ошибок и упрощает проект.

1. В библиотеке компонентов во вкладке Проект щелкните ПКМ на Типы → Создать. Откроется Мастер создания элементов. В поле Тип элемента выберите Тип, в поле Базовый тип выберите визуальный компонент Прямоугольник.

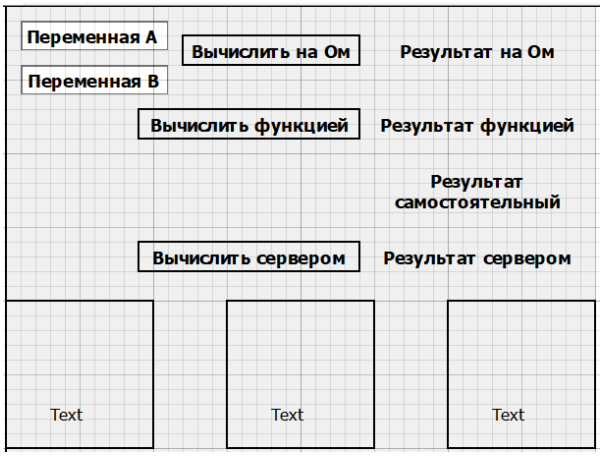


Графические типы можно создавать на основе любого графического элемента.

2. Перейдите внутрь созданного типа (Rectangle_1) двойным нажатием мыши.
3. Перетяните на тип **Rectangle_1** из библиотеки компонентов элемент ***Текст***.
4. Перейдите на форму **Form_1** и перетяните из Библиотеки компонентов 3 экземпляра типа **Rectangle_1**.

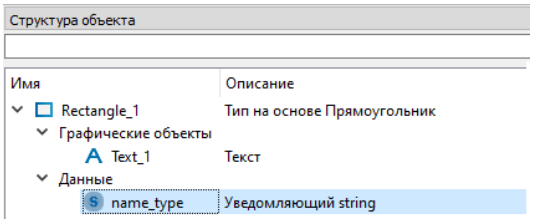


5. Перейдите в тип **Rectangle_1** и перетяните элемент ***Текст*** в нижнюю часть прямоугольника.
6. Перейдите на форму **Form_1**. Все экземпляры типа приняли эти изменения.

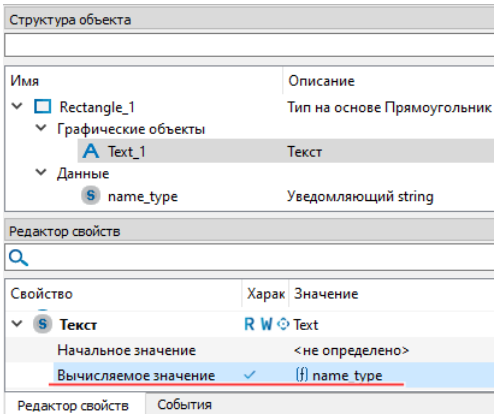


Изменения внутри типа влечёт за собой изменения у всех экземпляров. Но каждый экземпляр можно сделать уникальным. Можно переопределить свойство родительского элемента, на основе которого построен логический тип. Из этого следует следующая особенность типизации – возможность указывать какие-то уникальные свойства. Например, можно сделать так, чтобы у каждого экземпляра в текстовом поле размещалось своё уникальное название.

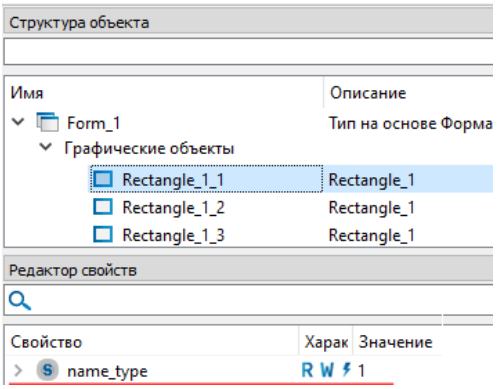
7. Перейдите к типу **Rectangle_1**. Создайте здесь ***Уведомляющее поле***: щелкните ПКМ по ***Rectangle_1*** в Структуре объекта → Создать. Тип элемента – Уведомляющее поле, Базовый тип элемента Уведомляющий string. Назовите его name_type.



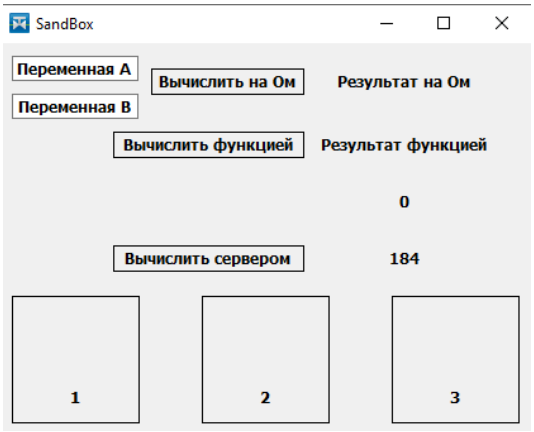
8. Выделите элемент ***Text_1*** и в Вычисляемое значение его свойства Текст поместите уведомляющее поле name_type.



9. Перейдите на форму **Form_1**, в структуре элемента выделите один из типов ***Rectangle_1***. В его свойстве name_type введите 1. То же сделайте и для оставшихся экземпляров.



10.Сохраните проект, запустите.

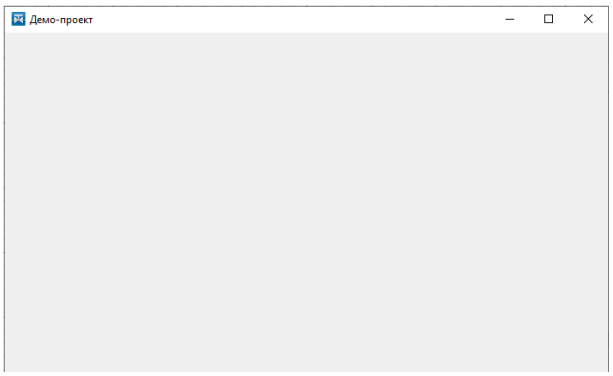


Теперь типовые элементы имеют своё индивидуальное имя. Это свойство поможем в наименовании всех датчиков и задвижек, которые будут располагаться на мнемосхеме в демонстрационном проекте.

Создание демонстрационного проекта в SePlatform.HMI

В демонстрационном проекте будут и датчики, и задвижки. Все экземпляры будут располагаться на форме **MainForm**.

1. Откройте форму **MainForm** двойным нажатием мыши.
2. В Структуре объекта выделите ***MainForm*** и в свойствах Ширина и Высота укажите 700.
3. В свойстве Размеры окна укажите Автоподбор, Положение окна – По центру монитора.
4. В свойстве Заголовок окна введите Демо-проект.
5. В библиотеке компонентов во вкладке Проект щелкните ПКМ на форме **MainForm** → Установить форму главной.
6. Сохраните проект, запустите.

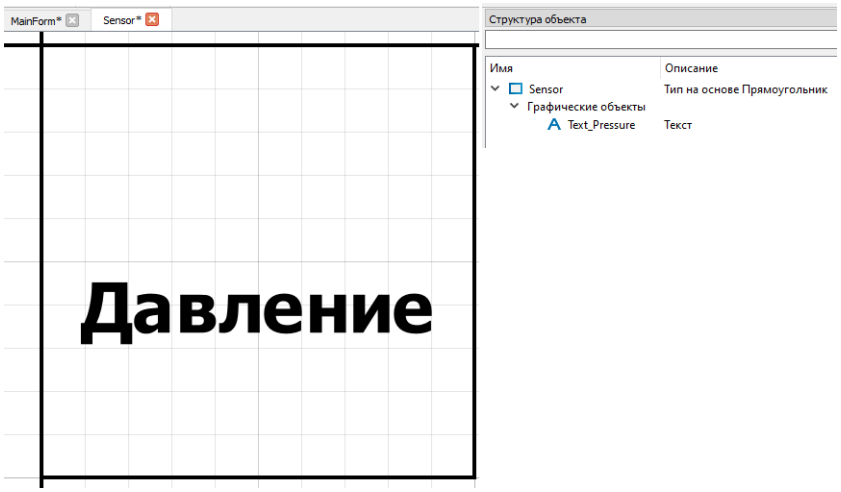


Перейдём к созданию типов датчика и задвижки.

Создание типа с датчиком

Сейчас наша задача вывести на главную форму экземпляры датчиков и их значения. Датчики будут располагаться внутри прямоугольника. Поэтому создадим новый тип на основе прямоугольника.

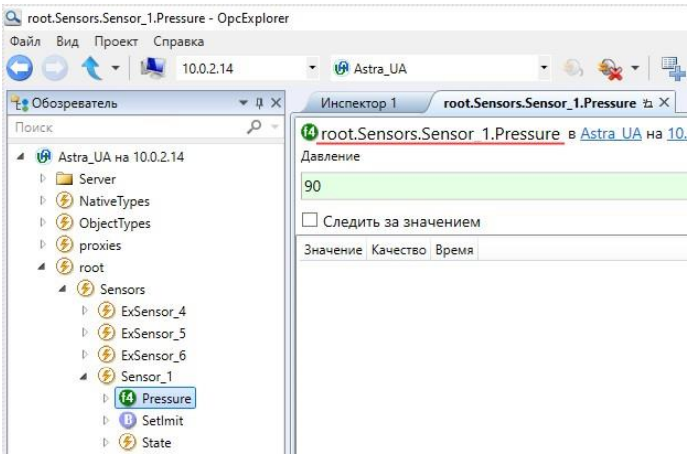
1. Создайте тип на основе Визуального компонента Прямоугольник, назовите его Sensor. Перейдите в него.
2. Перетяните из библиотеки компонентов элемент ***Текст***, назовите его Text_Pressure (в нём в дальнейшем будет отображаться значение давление).
3. В свойстве Текс этого элемента введите Давление. Отредактируйте шрифт и положение текста.



4. Перейдите к форме ***MainForm*** и разместите на ней 3 экземпляра **Типа Sensor**.

Теперь необходимо на каждый экземпляр датчика вывести значение давления, то есть нужно оформить подписку на сервер для того, чтобы он подписался на конкретные тэги датчика и вывел значение уже в RunTime. Но сначала давайте посмотрим, как должен выглядеть путь до датчика.

5. Перейдите в *OpcExplorer* и в Обозревателе кликните ЛКМ на сигнал ***Pressure*** внутри ***Sensor_1***.



Путь до конкретного датчика будем разбивать на 4 части:

root.Sensors.Sensor_1.Pressure
root.Sensors.Sensor_2.Pressure
root.Sensors.Sensor_3.Pressure

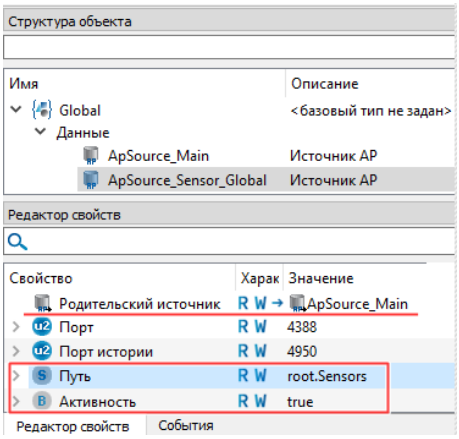
1. Точка подключения – номер хоста и номера порта. Укажем в «ApSource_Main» (в «Global»)
2. Часть пути – «root.Sensors». Укажем в «Ap_source_Sensors» (в том же «Global»)
3. Уникальная часть имени экземпляра. Укажем в «ApSource_Sensor_Type» (уже в самом типе) и добавим уведомляющее поле «_Sensor_Path» для индивидуальной настройки имени для каждого экземпляра «Sensor_Type»
4. Элемент Ap_Pressure + Pressure

1) Точка подключения (хост и порт). Эта точка подключение уже существует, ***ApSource_Main*** (***Global***).
2) Следующая часть пути повторяется абсолютно у всех датчиков:

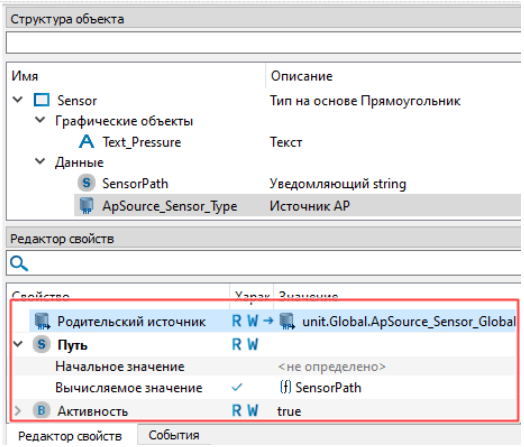
root.Sensors. Эту часть пути поместим в ***Источник АР***, назовём его ApSource_Sensors_Global и тоже разместим его в глобальных объектах (***Global***). Его особенностью будет то, что он от
предыдущего источника будет получать часть пути, в которой описаны хост и порт. И будет описывать свою часть пути: **root.Sensors**.

3) Следующая часть пути – это название конкретного датчика: Sensor_1, Sensor_2, Sensor_3. То есть как раз то, что отвечает за уникальность. Эту уникальную часть имени (SensorPath, будет храниться в уведомляющем поле) стоит создать внутри графического типа и разместить её в ***ApSource_SensorType*** (в вычисляемом значении свойства Путь). Особенность этого источника в том, что он будет наследовать то, что есть у других источников (то есть то, что есть у глобального источника и будет получать часть путь root.Sensors).
4) После этого создадим ***Элемент АР*** Ap_Pressure с оставшейся частью пути, которая повторяется везде. Этот элемент будет наследовать то, что было в предыдущих частях каждого конкретного датчика, но добавит оставшуюся часть пути – Pressure.

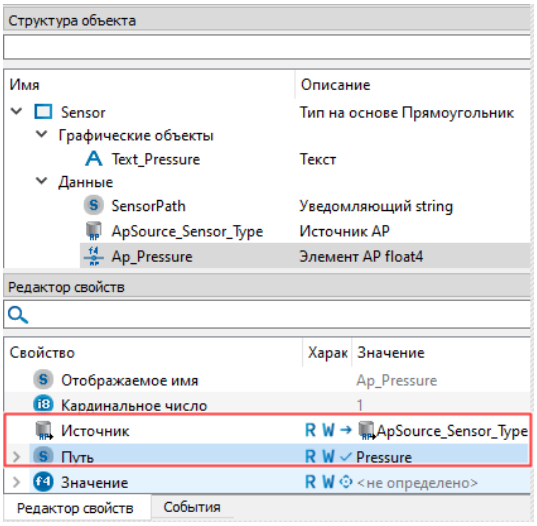
6. Откройте *SePlatform.HMI*, перейдите в глобальный объект ***Global***. Перетяните сюда из Библиотеки компонентов ***Источник АР***. Назовите его ApSource_Sensors_Global.
7. В Структуре объекта выделите ***Источник ApSource_Sensors_Global*** и в свойстве Родительский источник сошлитесь на ApSource_Main. Установите ему Активность true. В свойстве Путь введите одинаковую часть пути: root.Sensors.



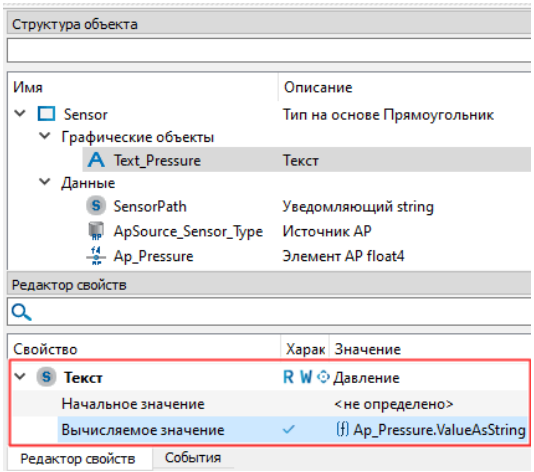
8. Перейдите в ***Тип Sensor***. Создайте здесь ***Уведомляющее поле*** типа string, назовите его SensorPath (здесь будет храниться индивидуальная часть пути Sensor_1, Sensor_2 и Sensor_3).
9. Перетяните из Библиотеки компонентов на Тип Sensor ***Источник АР***, назовите его ApSource_SensorType. В свойстве Родительский источник сошлитесь на ApSource_Sensors_Global, находящийся в глобальных объектах. В вычисляемое значение свойства Путь поместите SensorPath, Активность: true.



10.Перетяните из Библиотеки компонентов на тип Sensor ***элемент Ap*** float4, назовите его Ap_Pressure. В свойстве Источник сошлитесь на ApSource_SensorType, в свойстве Путь укажите оставшуюся часть – Pressure.

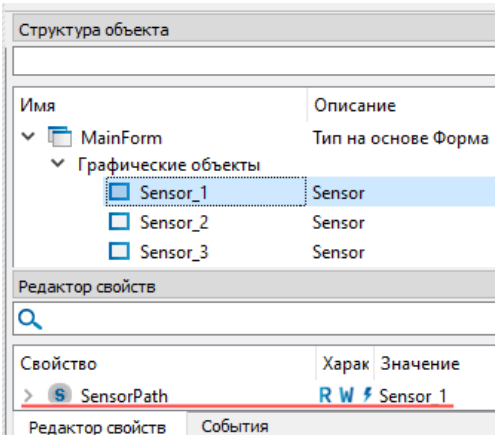


11.В Структуре объекта выделите элемент ***Text_Pressure***, и в Вычисляемое значение его свойства текст поместите значение Ap_Pressure в текстовом формате: Ap_Pressure.ValueAsString.

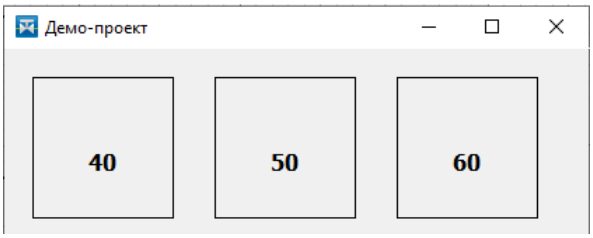


Первая часть создания типа датчика готова, можно переходить к редактированию экземпляров.

12.Перейдите в экранную форму **MainForm**, выделите в Структуре объекта ***Графический объект Sensor_1***. В свойстве SensorPath укажите индивидуальную часть пути: Sensor_1.



13.Повторите пункт 12 для ***Sensor_2*** и ***Sensor_3***.
14.Сохраните проект, запустите форму в RunTime.

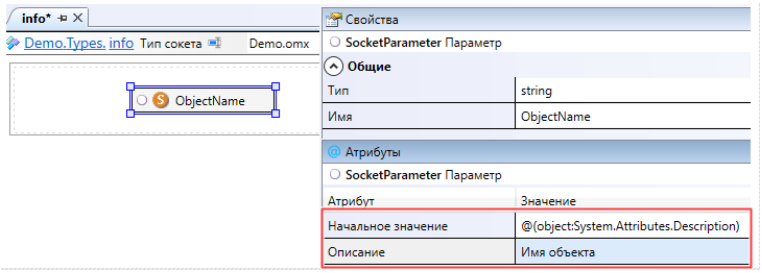


Таким образом мы пробросили значение ***Параметра Pressure*** на экземпляры датчика в *Alha.HMI*. Осталось пробросить имя каждого датчика, которое хранится внутри атрибута Описание у объектов.

15.Перейдите в ***Тип Sensor*** и перетяните сюда из Библиотеки компонентов визуальный элемент ***Текст***. Назовите его Text_Name. В его свойстве Текс введите Имя. Настройте шрифт и выравнивание текста.

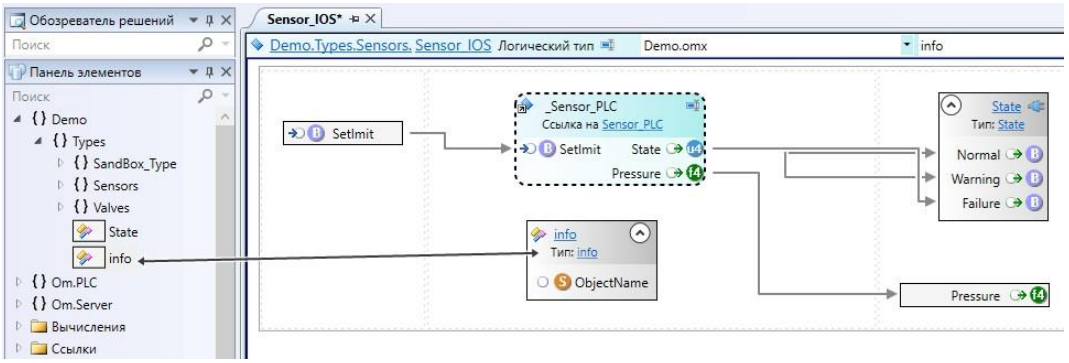
Для того, чтобы вывести имя, нужно каким-то образом вытащить описание у объекта, внутри которого находятся наши датчики. Но *SePlatform.HMI* не умеет подписываться на объекты. Он умеет
подписываться только на конкретные сигналы. Соответственно свойства он может вытащить только у сигналов. Мы можем сделать в сервере дополнительный тэг, который будет хранить в себе имя объекта, внутри которого он содержится. Для этого нам нужно будет немного модифицировать проект в *DeveloperStudio*.

16.Откройте *DevStudio*, перейдите в Пространство имён с типами ***Types*** и добавьте сюда из Панели элементов ***Тип сокета***. Назовите его info.
17.Перейдите в ***Сокет info*** и перетяните сюда из Панели элементов ***Параметр*** типа string. Назовите его ObjectName.
18.Добавьте ***Параметру ObjectName*** атрибуты Описание и Начальное значение. В атрибуте Описание введите Имя объекта, в атрибуте Начальное значение - **@(object:System.Attributes.Description)** (это обращение к объекту, внутри которого будет содержаться данный параметр и обращение по пути к атрибуту). Таким образом мы вытащили стандартный атрибут System.Attributes.Description.



Теперь необходимо разместить этот ***Тип сокета*** в типах датчика и задвижки со стороны Сервера ввода-вывода.

19.Перейдите в ***Types*** → ***Sensors*** → ***Sensor_IOS*** и из Панели элементов перетяните сюда ***Тип сокета*** info.



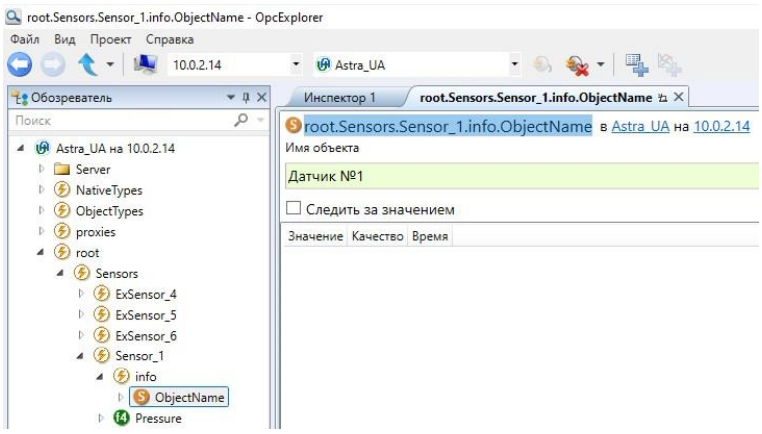
20.Перейдите в ***Types*** → ***Valves*** → ***Valve_IOS*** и из Панели элементов перетяните сюда ***Тип сокета info***.

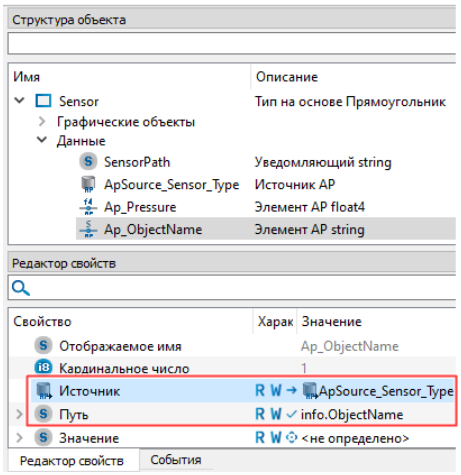
21.**Постройте решение**, **перейдите к развёртыванию** и **залейте конфигурацию** на ***AstraServer***.

Теперь необходимо подписаться на этот строковый элемент у датчика в *SePlatform.HMI*.

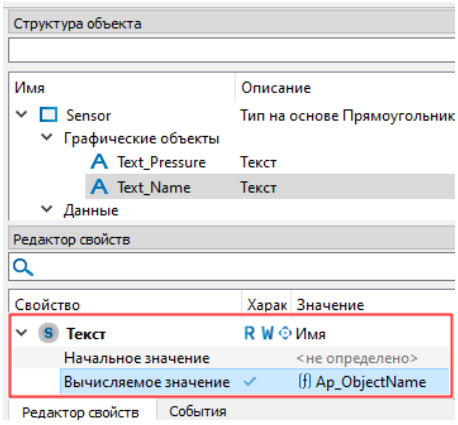
22.Откройте проект в *SePlatform.HMI*, перейдите в ***Тип Sensor***. Из библиотеки компонентов перетяните сюда ***Элемент АР*** string, назовите его Ap_ObjectName.

23.В структуре объекте выделите ***Ap_ObjectName***. В свойстве Источник сошлитесь на ApSource_SensorType, а в свойстве Путь то, чего еще не хватает: info.ObjectName (полный тэг до сигнала можно посмотреть в *OpcExplorer*).



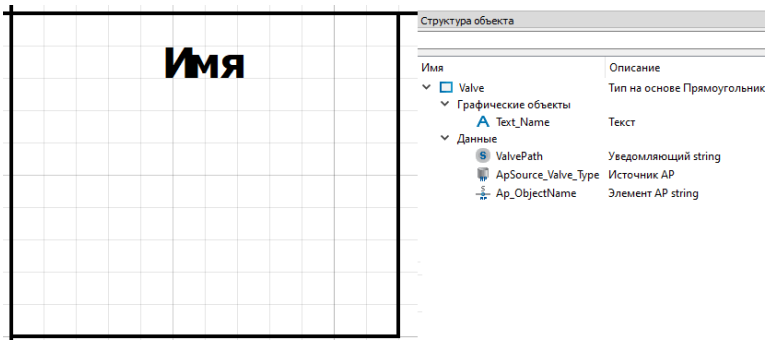


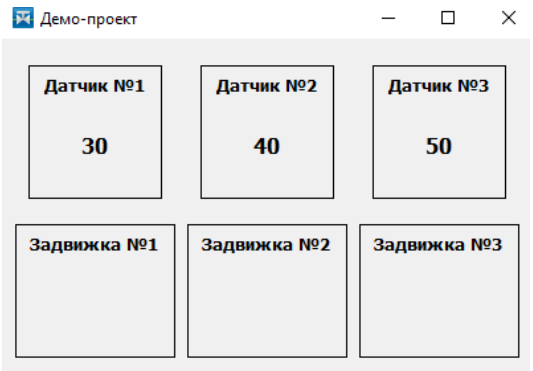
24.В структуре объекта выделите элемент ***Text_Name*** и в Вычисляемое значение его свойства Текст поместите Ap_ObjectName.



25.Сохраните проект, запустите главную форму в RunTime. Теперь все датчики подписаны и отображают значение давления.

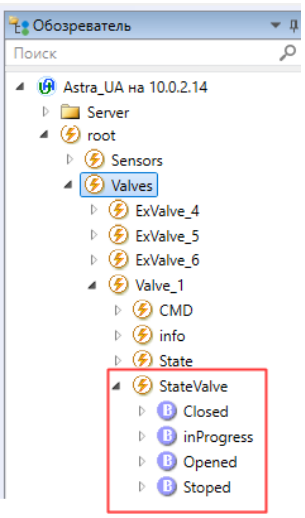
Самостоятельное задание. Создайте ***Тип Valve***, разместите на нём текстовое поле, в котором будет выводиться имя задвижки. Разместите 3 экземпляра типа на форме **MainForm**.



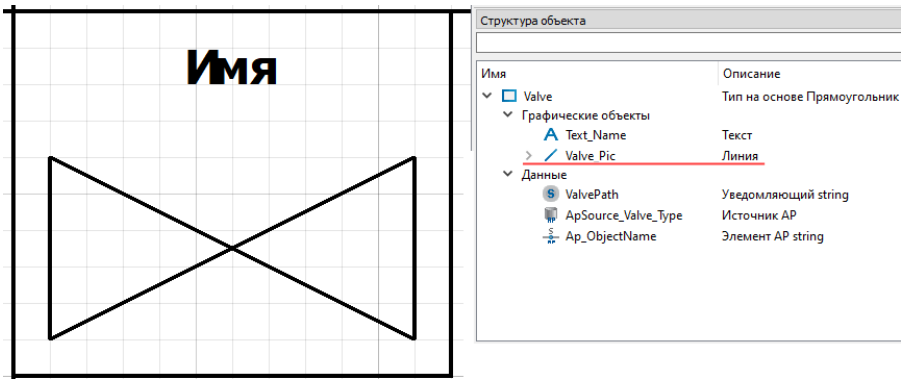


Добавление цветовой индикации

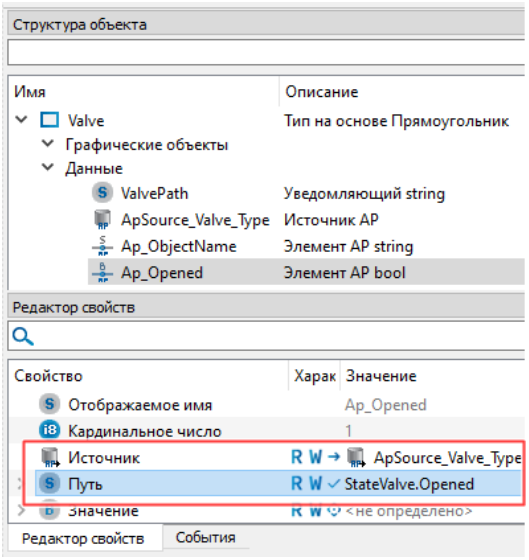
У задвижки есть 4 состояние: Opened, Closed, Stoped, inProgress (сокет StateValve). Наша задача: нарисовать на ***Типе Valve*** задвижку, подписаться на эти состояния и в зависимости от состояния закрашивать задвижку.



1. В *SePlatform.HMI* перейдите в ***Тип Valve***. Из Библиотеки компонентов (раздел Визуальные) перетяните элемент ***Линия*** и нарисуйте с её помощью задвижку. В структуре объекта назовите эту линию Valve_Pic.



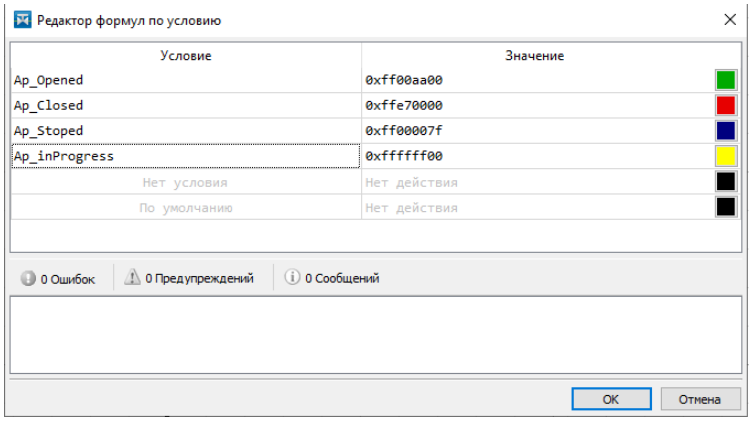
2. В структуре объекта выделите ***Линию Valve_Pic***, в свойстве Стиль заливки выберите Сплошная заливка. Задвижка окрасится в чёрный цвет.
3. Из библиотеки компонентов (раздел AP) перетяните ***Элемент AP*** bool, назовите его Ap_Opened.
4. В Структуре объекта выделите ***Ap_Opened***, в свойстве Источник сошлитесь на ApSource_Valve_Type, в свойстве путь укажите недостающую часть пути: **StateValve.Opened**.



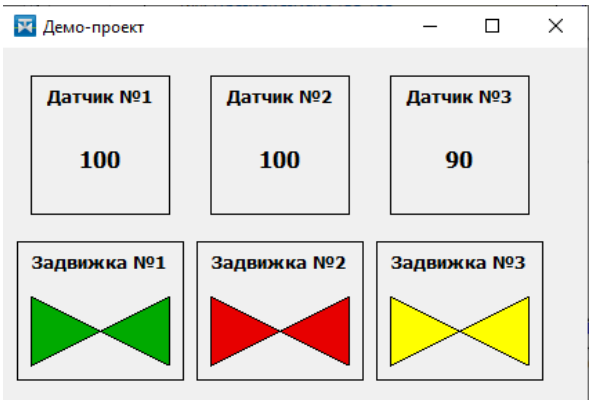
5. Повторите пункты 3 и 4 для ***Ap_Closed***, ***Ap_Stoped*** и ***Ap_inProgress***.

Теперь изменим цвет заливки задвижки.

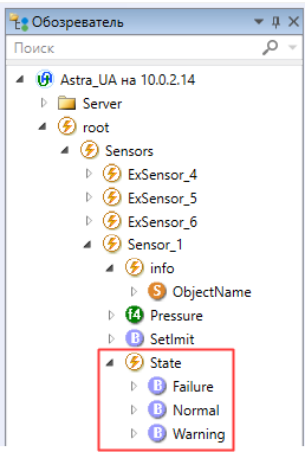
6. В структуре объекта выделите элемент ***Valve_Pic***, перейдите в свойство Цвет заливки → Вычисляемое значение → Формула по условию → Редактировать. Откроется Редактор формул по условию. Введите значения в соответствии с изображением.

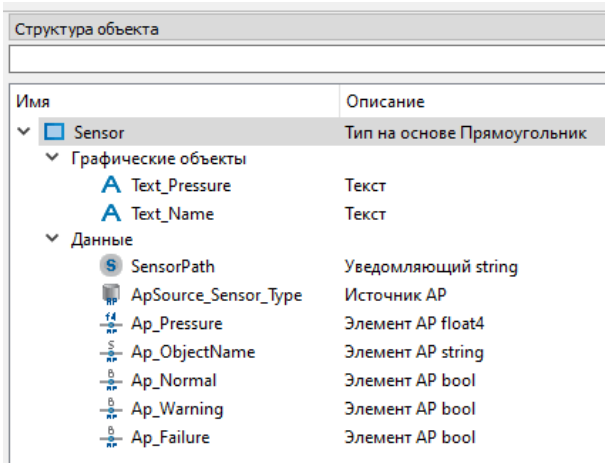


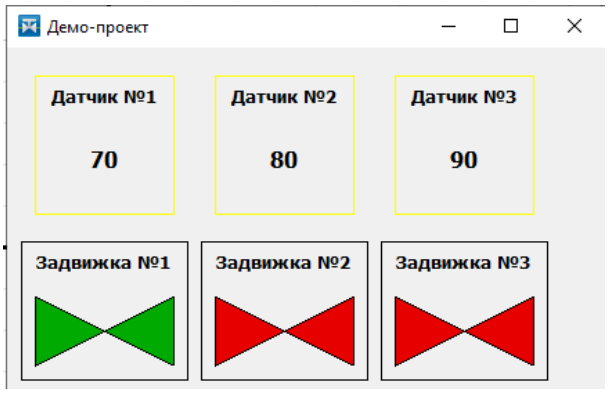
7. Сохраните проект, запустите главную форму в RunTime. Теперь задвижки меняют свой цвет в зависимости от состояния.



Самостоятельное задание. У каждого датчика есть набор состояний state: normal, warning, failure. В *SePlatform.HMI* у датчика есть контур прямоугольника (в свойствах он называется пером). Нужно, чтобы контур каждого прямоугольника окрашивался в цвет в зависимости от состояния: normal – зелёный, warning – жёлтый, failure – красный. То есть нужно будет подписаться на дополнительные булевские параметры и сделать окрашивание у датчиков.







Открытие форм через обработчик

Следующая задача – это создание формы управления для каждого датчика и каждой задвижки. У задвижек есть команды управления: open, close, stop. Кнопки управления будут размещены на отдельной форме, через которую в дальнейшем будет осуществляться управление задвижкой. Для управления датчиками тоже создадим отдельную форму, где будет отображено текущее значение, состояние и имя датчика. Также здесь будет размещён флажок, позволяющий запускать и останавливать имитацию.

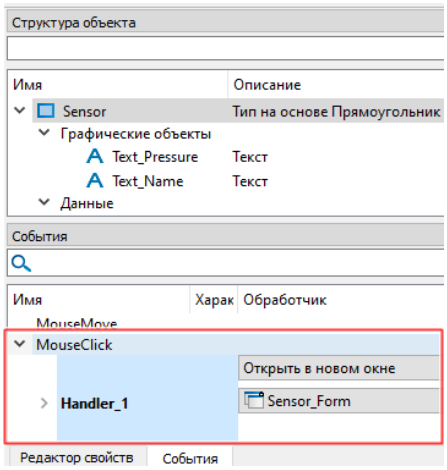
Для начала создадим новую экранную форму.

1. В библиотеке компонентов (вкладка Проект) щелкните ПКМ по Экранные формы → Создать. Откроется мастер создания элементов. В поле Тип элемента оставьте Тип, Базовый тип элемента – Форма (раздел Визуальные). Назовите эту форму Sensor_Form.
2. Перейдите в форму **Sensor_Form**, в Структуре объекта выделите форму, перейдите к её свойствам. Заголовок окна: Управление датчиком, Размеры окна: Автоподбор, Положение окна: По центру монитора.

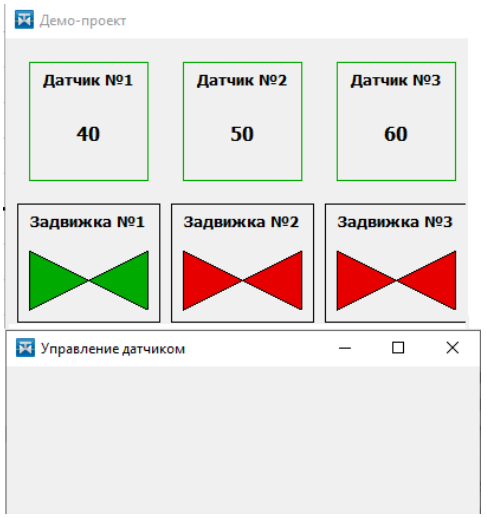
Эта форма будет открываться в тот момент, когда произойдёт нажатие мыши по любому датчику. Давайте это настроим.

3. Перейдите в **Графический тип Sensor**, в Структуре объекта выделите прямоугольник ***Sensor***, перейдите во вкладку События.

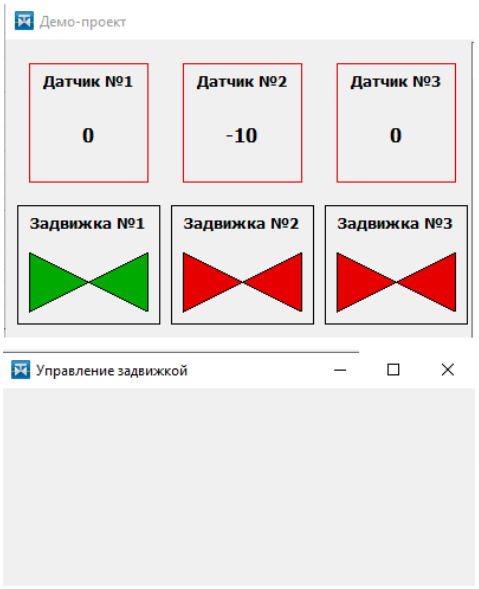
4. Щелкните ПКМ по событию MouseClick → Добавить обработчик → Открыть в новом окне → **Sensor_Form**.



5. Сохраните проект, запустите главную форму в RunTime. При нажатии на любой из датчиков открывается форму управления.



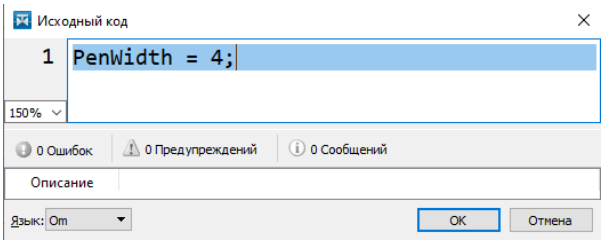
Самостоятельно задание. Создайте новую фурму управления задвижкой **Valve_Form**, задайте все необходимые свойства и настройте открытие этой формы при нажатии на любую из задвижек на главной форме.



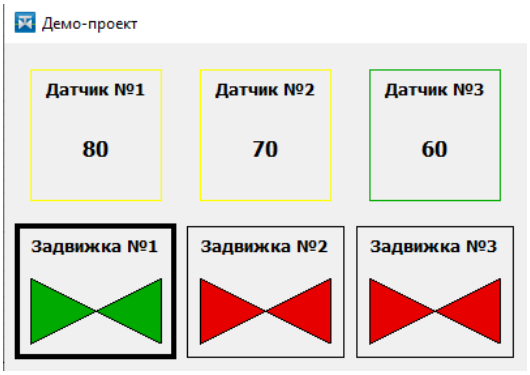
Добавление анимации

Было бы хорошо дать понять пользователю, что элемент интерактивный, и с ним можно взаимодействовать. Обычно это значит, что нужно добавить какую-то анимацию, например, выделение контура элемента.

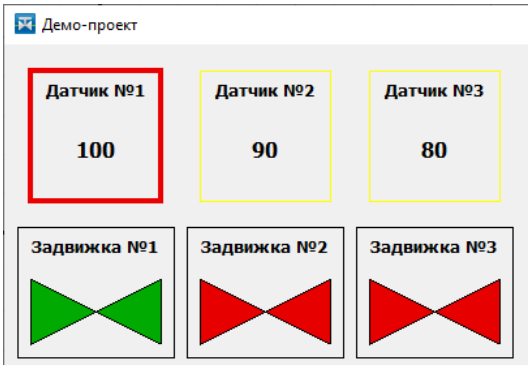
1. Перейдите в **Графический тип Sensor**. В структуре объекта выделите прямоугольник ***Sensor***, перейдите во вкладку События.
2. Щелкните ПКМ по событию MouseEnter → Добавить обработчик → Выполнить код → Редактировать. Откроется окно Исходный код. Впишите сюда код в соответствии с изображением (свойство PenWidth отвечает за толщину пера).



3. При срабатывании события MouseLeave создайте обработчик, который приводит толщину пера к состоянию, равному 1.
4. Сохраните проект, запустите главную форму в RunTime. При наведении курсора на любую из задвижек контур становится жирнее.



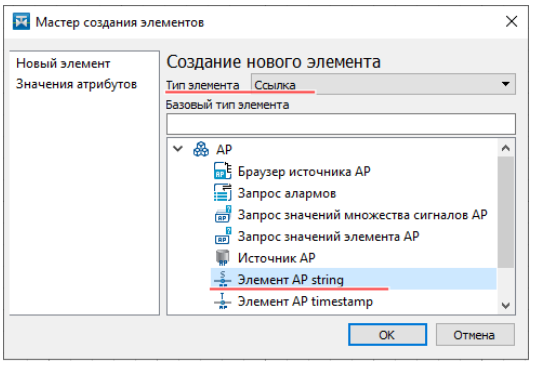
Самостоятельное задание. С делайте то же самое для датчиков: при наведении мыши на прямоугольник контур должен становиться жирнее.



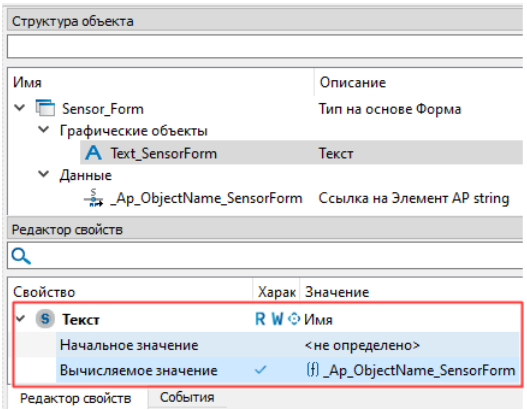
Создание ссылок на основе примитива

Теперь необходимо в эти дочерние формы передать имена датчиков и задвижек. Эти имена у нас уже проброшены в типах **Sensor** и **Valve**, поэтому необходимо сослаться на элемент ***Ap_ObjectName***, который и отвечает за имя объекта. Имя в дочерних формах **Sensor_Form** и **Valve_Form** будет размещаться в текстовом поле.

1. Перейдите на экранную форму **Sensor_Form**. Перетяните сюда из Библиотеки копмонентов элемент ***Текст***, назовите его Text_SensorForm. Пропишите внутри «Имя», настройте шрифт и положение текста.
2. Для того, чтобы создать ссылку в Структуре объекта нажмите ПКМ по ***Sensor_Form*** → Создать. Откроется Мастер создания элементов. В поле Тип элемента выберите Ссылка, Базовый тип – Элемент AP string. В Структуре объекта назовите её _Ap_ObjectName_SensorForm.

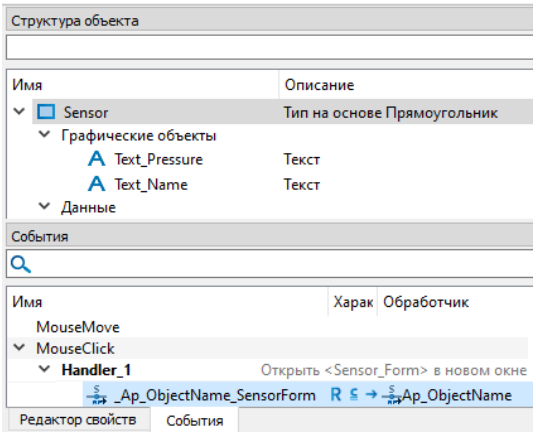


3. В Структуре объекта выделите элемент ***Text_SensorForm***, в Вычисляемое значение его свойства Текст поместите _Ap_ObjectName_SensorForm

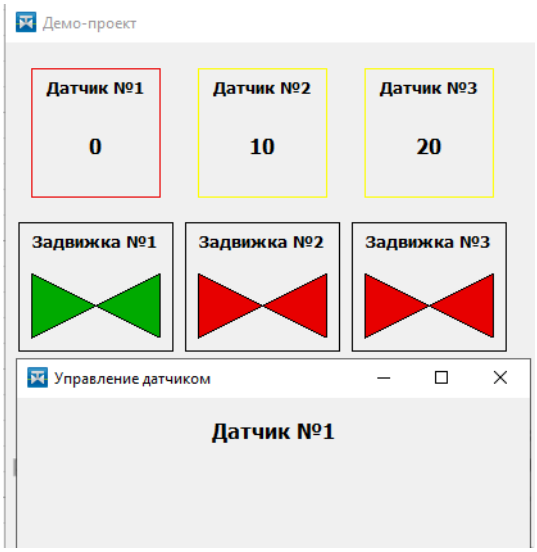


Теперь эту ссылку необходимо проинициализировать. Инициализация родительского и дочернего контекста, то есть момент соприкосновения родительского и дочернего контекста, происходит в обработчиках. Форма ***SensorForm*** открывается тогда, когда мы кликаем на датчик. То есть, когда срабатывает событие MouseClick.

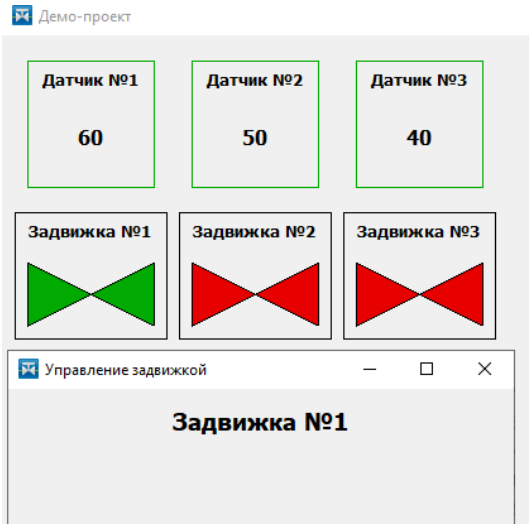
4. Перейдите в тип **Sensor**. В Структуре объекта кликните на прямоугольник ***Sensor*** и откройте вкладку События. Раскройте список MouseClick, раскройте список Handler. В конце списка есть как раз та самая ссылка, здесь её и нужно проинициализировать. Нажмите ПКМ по значению свойства _Ap_ObjectName_SensorForm → Сослаться → Ap_ObjectName.



5. Сохраните проект, запустите главную форму в RunTime. При нажатии на любой из датчиков открывается форма управления с соответствующей надписью.



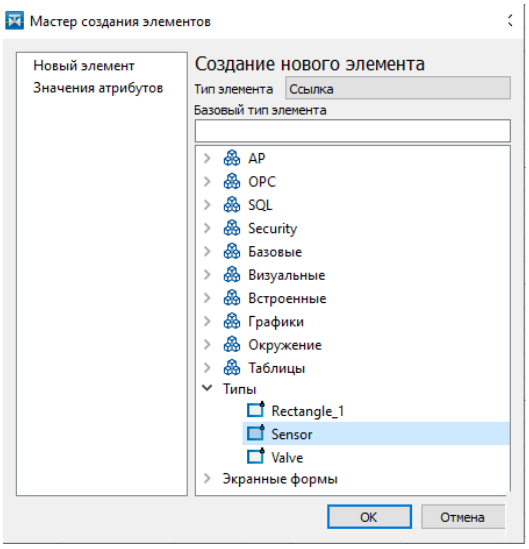
Самостоятельное задание. Пробросьте на форму управления задвижек имя соответствующей задвижки.



Это один из способов работы со ссылкой. Давайте попробуем сейчас вывести значение датчика в текстовом поле.

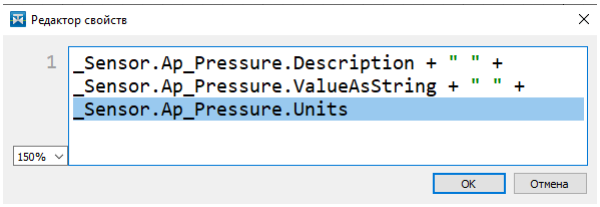
Создание ссылок на основе графического типа

1. Перейдите в форму **SensorForm**. Добавьте сюда элемент ***Текст*** с надписью Значение датчика. Назовите его Text_Sensor_Value. Настройте шрифт, положение текста.
2. Для того, чтобы создать ссылку на графический тип, нажмите в Структуре объекта ПКМ на Sensor_Form → Создать. Тип элемента: Ссылка, Базовый тип элемента: Типы → Sensor. Назовите её _Sensor.



Добавление этой ссылки означает, что мы уже можем взаимодействовать с теми элементами, которые есть внутри графического типа.

3. В Структуре объекта выделите ***Text_Sensor_Value*** и в Вычисляемом значении его свойства Текст введите скрипт в соответствии с изображением.

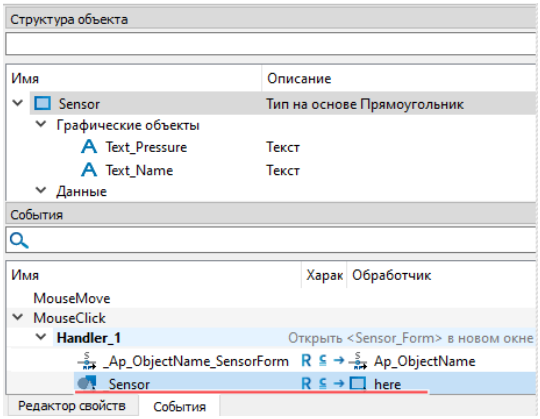


То есть выведем описание, значение в текстовом формате и единицы измерения элемента ***Ap_Pressure***.

Таким образом можно воспользоваться любым из параметров, который есть внутри типа, к которому ведёт ссылка.

Теперь осталось только проинициализировать эту ссылку.

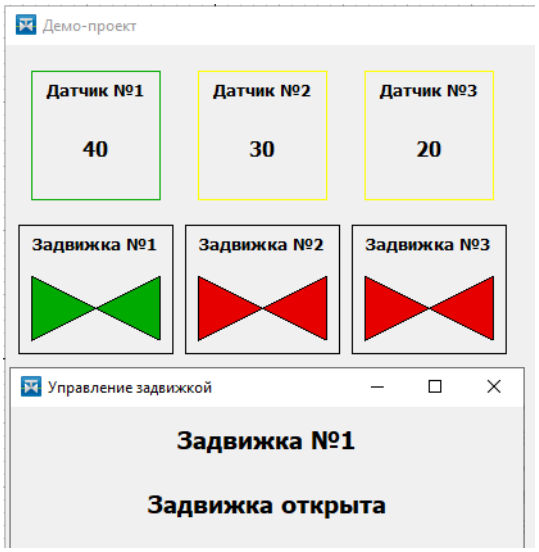
4. Перейдите в тип **Sensor**. В Структуре объекта выделите прямоугольник ***Sensor***, перейдите к его событиям. Раскройте список события MouseClick, раскройте список обработчика Handler, перейдите к пункту _Sensor, нажмите ПКМ → Сослаться → here (here – специальное слово для того, чтобы ссылаться на весь тип).



5. Сохраните проект, запустите главную форму в RunTime. При нажатии на любой из датчиков откроется форма управления датчиком с соответствующими именем и значением датчика.

Самостоятельное задание. У задвижек есть 4 булевских параметра: Opened, Closed, Stoped и inProgress. На форме **Valve_Form** необходимо разместить ***элемент Текст*** и в это поле необходимо

поместить описание состояния задвижки: в зависимости от того, у какого из булевских параметров значение true, то описание и выводить. При выполнении использовать ссылку на графический тип **Valve**.

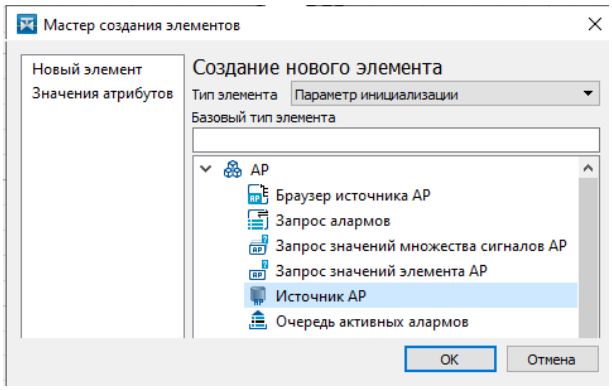


Работа с параметром инициализации

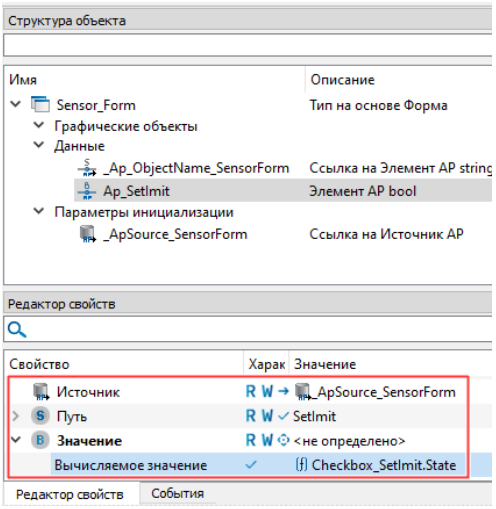
У датчиков и задвижек есть сигналы, через которые можно отправлять различные управляющие воздействия. Например, в случае с датчиком это ***сигнал SetImit***, которому можно установить либо true, либо false. Если отправите true, то имитация будет продолжаться, если false, то имитация остановится. У задвижек такими командами являются Open, Close, Stop.

Управляющие воздействия будут отправляться с форм управления. Но для этого нужно пробросить путь до источника. Для того, чтобы создать ссылку на источник, где уже будет путь до определённого объекта, нужно создать ***Параметр инициализации***.

1. Перейдите в форму **Sensor_Form**.
2. Для того, чтобы создать ***Параметр инициализации***, щелкните ПКМ на ***Sensor_Form*** в Структуре объекта → Создать. Откроется Мастер создания элементов. В поле Тип элемента выберите Параметр инициализации, в поле Базовый тип элемента – AP → Источник AP. Назовите его _ApSource_SensorForm.

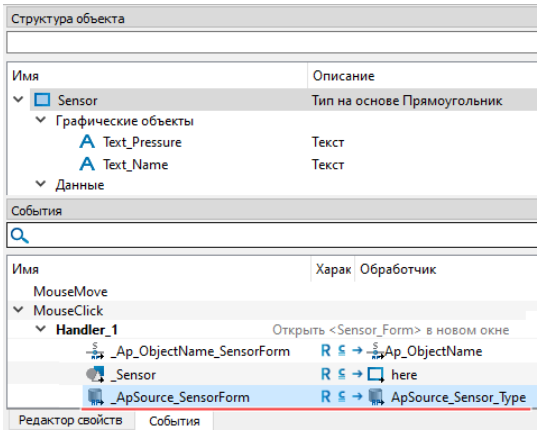


3. Перетяните из Библиотеки компонентов ***элемент AP*** bool, назовите его Ap_SetImit.
4. Выделите ***Ap_SetImit*** в Структуре объекта, в свойстве Источник сошлитесь на _ApSource_SensorForm, а в свойстве Путь укажите оставшуюся часть пути: SetImit.
5. Перетяните из Библиотеки компонентов элемент ***Флажок*** (раздел Визуальные), назовите его CheckBox_SetImit. В свойстве Текст введите Управление имитацией. Настройте шрифт.
6. В Структуре объекта выделите ***Ap_SetImit***, в Вычисляемом значении его свойства Значение поместите **CheckBox_SetImit.State** (то есть меняется State и сразу отправляется в сервер).

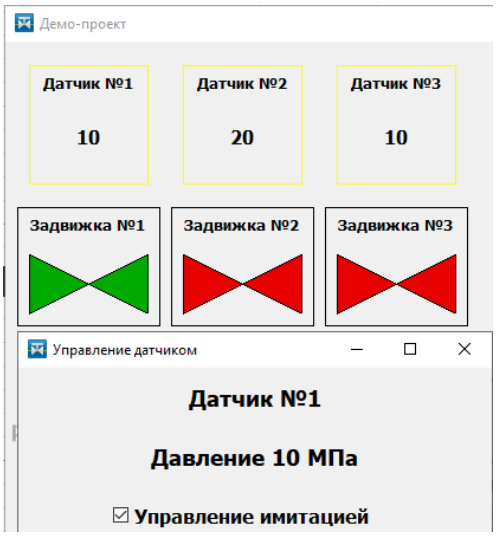


Теперь нужно проинициализировать эту ссылку. Инициализация также происходит в момент соприкосновения родительского и дочернего контекста, а этой точкой соприкосновения является момент открытия формы.

7. Перейдите к типу **Sensor**. В Структуре объекта выделите прямоугольник ***Sensor***, перейдите во вкладку События. Раскройте список события MouseClick, раскройте список обработчика Handler, перейдите к пункту _ApSource_SensorForm, нажмите ПКМ → Сослаться → ApSource_Sensor_Type.



8. Сохраните проект, запустите главную форму в RunTime. При нажатии на любой из датчиков открывается форма управления, где можно управлять датчиком.



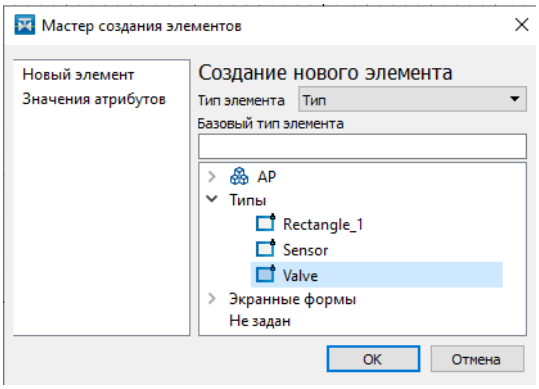
Самостоятельное задание. На форме ***Valve_Form*** создать кнопки для управления задвижкой. Подписаться на 3 команды управления задвижкой. При нажатии на кнопку отправлять значение true.

Наследование

В демонстрационном проекте помимо простых датчиков и задвижек имеются ещё расширенные датчики и задвижки. Расширенные датчики предоставляют информацию об уставках, а расширенные задвижки предоставляют информацию о проценте открытия. Их мы будем создавать, как и в *DeveloperStudio*, на основе типов **Sensor** и **Valve**.

SePlatform.HMI позволяет разрабатывать графические типы не только на основе графических примитивов, но и на основе других элементов, которые уже существуют в проекте. То есть можно создать графический тип на основе другого типа, и это будет считаться наследованием.

1. Для создания расширенного типа задвижки в Библиотеке компонентов (вкладка Проект) кликните ПКМ по Типы → Создать. Откроется окно Мастер создания элементов. В поле Тип элемента выберите Тип, в поле Базовый тип элемента – Типы → Valve. Назовите его ExValve.



Добавим к расширенному типу ***Индикатор прогресса***, который будет отображаться в проценте открытия задвижки. Но перед этим разместим экземпляры этого типа на главную форму

2. Перейдите в форму **MainForm**. Разместите здесь 3 экземпляра типа **ExValve**.
3. Выделите в Структуре объекта ***ExValve_1*** и в свойстве ValvePath укажите оставшуюся часть пути до этой задвижки: ExValve_4. Проделайте то же самое и с ***ExValve_2*** и ***ExValve_3***.
4. Сохраните проект, запустите главную форму в RunTime.

Так как расширенные задвижки созданы на основе простых, на них уже проброшены имя и обработчик открытия формы управления

Самостоятельное задание. Создайте расширенный тип датчика и также разместите 3 экземпляра на главной форме.

5. Перейдите к типу **ExValve**. Из Библиотеки компонентов (раздел Визуальные) перетяните элемент ***Индикатор прогресса***. Назовите его ProgressBar_OpenPrc.

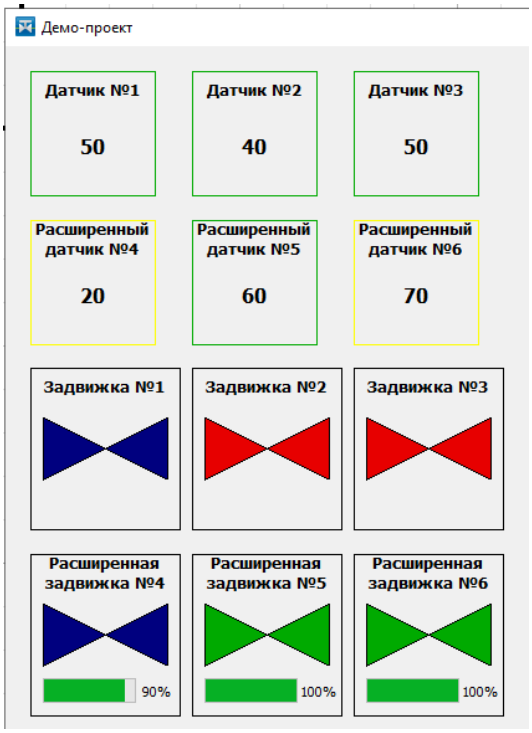
6. Из Библиотеки компонентов перетяните на тип **ExValve** ***Элемент АР*** uint4, назовите его Ap_OpenPrc.

У простой задвижки уже есть элемент ***ApSource***, то есть уже есть источник с путём до объекта. Значит у расширенной задвижки добавлять Источник не нужно, он как бы есть, но в структуре наследника он не отображается.

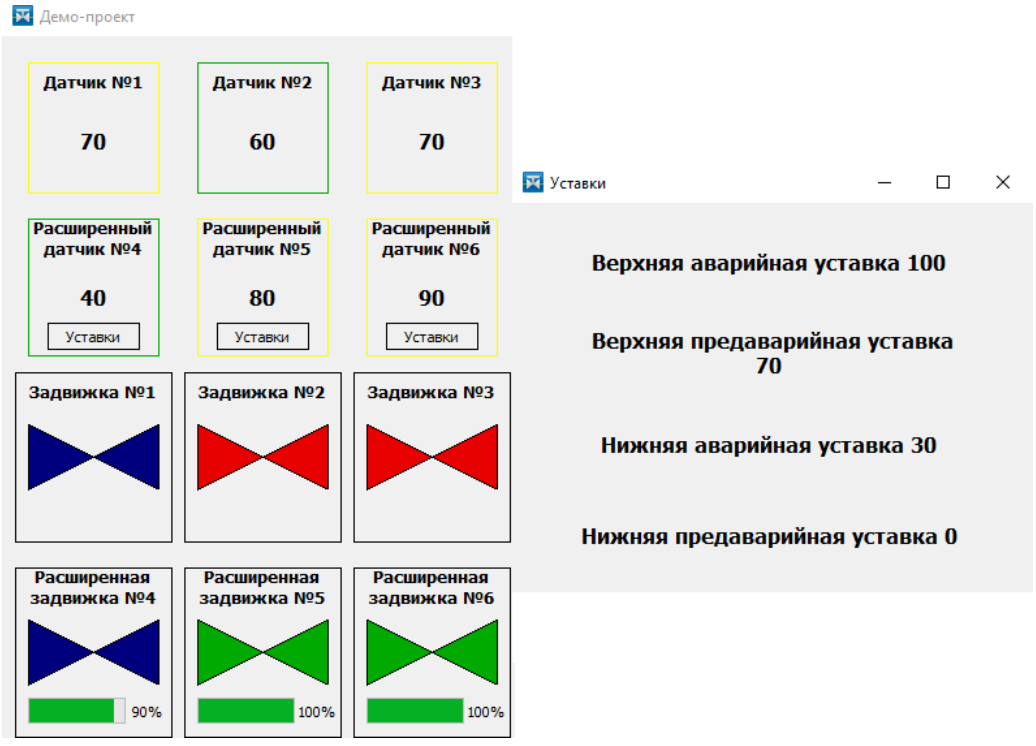
7. В Структуре объекта выделите ***Ap_OpenPrc***, в свойстве Источник сошлитесь на ApSource_Valve_Type, в свойстве Путь укажите недостающую часть пути: OpenPrc.

8. В Структуре объекта выделите ***ProgressBar_OpenPrc***, в Вычисляемом значении его свойства Значение введите Ap_OpenPrc.

9. Сохраните проект, запустите его в RunTime. ***Индикатор прогресса*** отображает процент открытия задвижки.



Самостоятельное задание. На форме **Sensor** необходимо разместить ***Кнопку***, при нажатии на которую должна открываться новая форма. В этой форме нужно отобразить текущее состояние уставок: описание, значение в текстовом формате и единицы измерения. Уставки хранятся внутри ***сокета Presets*** (HiHiLimit. HiLimit, LoLimit, LoLoLimit). При выполнении задания следует воспользоваться наследованием и передачей ссылки на источник через ***Параметр инициализации*** для того, чтобы подписаться на дополнительные тэги.



Сохраните проект, передайте его на машину с ОС Linux, запустите проект там (шрифты на ОС Windows и Linux могут отличаться).

Установка дополнительных библиотек и готовых решений в SePlatform.HMI

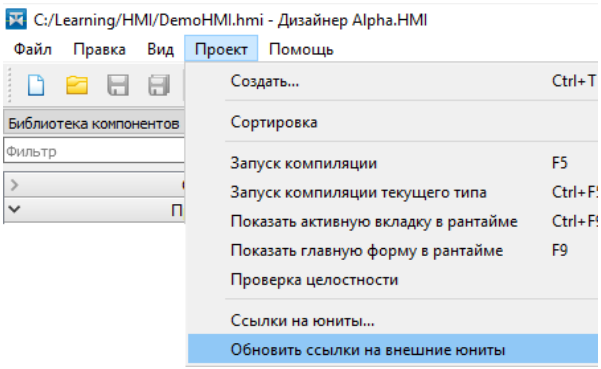
Перед тем, как устанавливать новые компоненты, закройте *SePlatform.HMI* на ОС Windows и ОС Linux (чтобы дополнительные библиотеки подтянулись в проект).

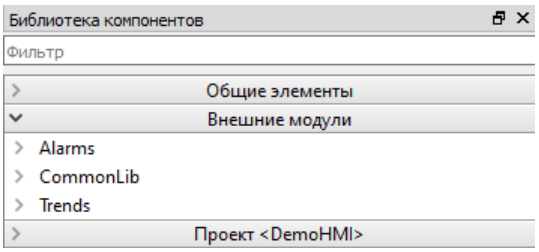
На ОС Windows установите внешние библиотеки и готовые решения:

- *SePlatform.hmi.charts* (для получения возможности представлять данные, принятые от источника, в виде графиков. Внешняя библиотека.);
- *SePlatform.hmi.tables* (для получения возможности помещать данные, полученные от источника по TCP, в таблицу и формировать собственные таблицы данных. Внешняя библиотека);
- *SePlatform.hmi.alarms* (компонент, выполненный в виде проекта *SePlatform.HMI*. Чтобы использовать *SePlatform.HMI.Alarms* в проекте автоматизации, нужно подключить его как внешний модуль. Для работы с *SePlatform.HMI.Alarms* понадобится также библиотека вспомогательных компонентов – внешний модуль *CommonLib*);
- *SePlatform.hmi.trends* (компонент, предназначенный для просмотра графиков изменений параметров технологического процесса. Выполнен в виде проекта *SePlatform.HMI*);
- *SePlatform.hmi.commonlib* (это расширение среды разработки и исполнения *SePlatform.HMI*. Расширение представляет собой библиотеку компонентов, которые можно использовать в своих проектах автоматизации: диалоговые окна, контекстное меню, календарь, файловый менеджер, компоненты безопасности).

Чтобы использовать *SePlatform.HMI.Alarms* и *SePlatform.HMI.Trends* в проекте автоматизации, нужно подключить их как внешние модули. Для этого:

1. Откройте 2 проводника: папка с проектом в *SePlatform.HMI* и C:\Program Files\SePlatform\SePlatform.HMI.Extensions.
2. В папке с проектом создадим папку с именем externals.
3. Скопируйте всё из папки C/Program Files\SePlatform\SePlatform.HMI.Extensions и вставьте в созданную папку externals.
4. Откройте проект в *SePlatform.HMI*, нажмите на вкладку Проект → Обновить ссылки на внешние юниты. После этого в Библиотеке компонентов появится новая вкладка Внешние модули с разделами Alarms, CommonLib и Trends.





Перейдём к установке внешних библиотек на ОС Linux. Здесь нужно установить только *SePlatform.hmi.charts* и *SePlatform.hmi.tables*.

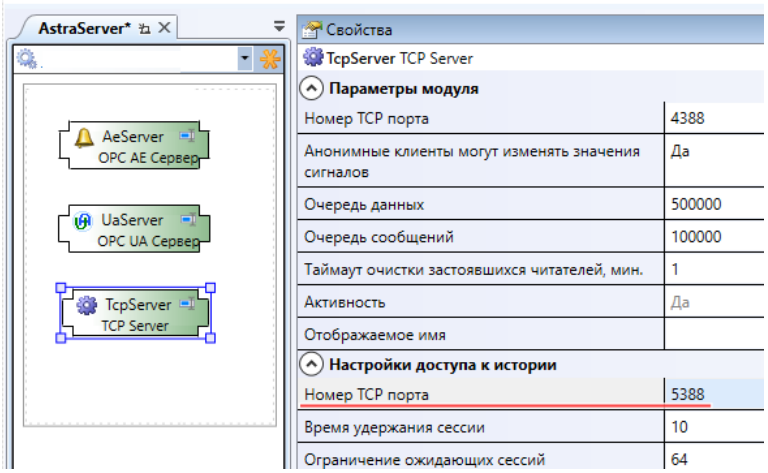
1. Откройте PuTTY Configuration, проверьте, что Вы находитесь в папке с дистрибутивами.
2. Для установки *SePlatform.hmi.charts* введите команду **sudo dpkg –i**

SePlatform.hmi.charts******.deb**.

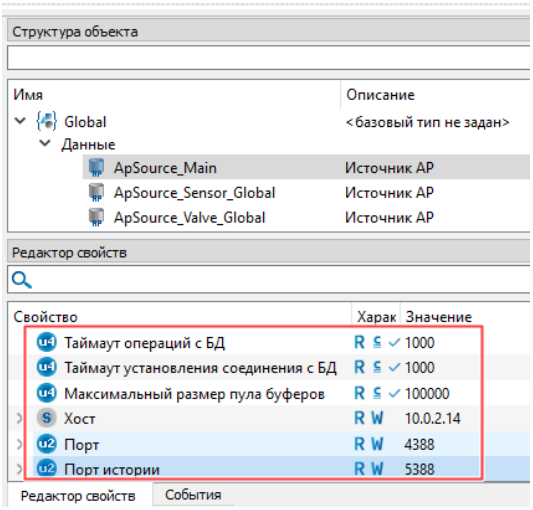
3. Для установки *SePlatform.hmi.tables* введите команду **sudo dpkg –i SePlatform.hmi.tables********.deb**.

Так как у нас в трендах и алармах будут использоваться исторические данные, давайте внесём небольшие изменения в проект *DeveloperStudio*.

1. Откройте *DevStudio*, перейдите в ***AstraServer***.
2. Выделите ***TcpServer***. В свойстве Номер TCP порта в разделе Настройки доступа к истории введите 5388 (для того, чтобы *SePlatform.HMI* высчитывал исторические данные).



3. **Постройте решение**, **перейдите к Мастеру развёртывания** и **примените конфигурацию к AstraServer**.
4. Откройте *SePlatform.HMI*, перейдите в глобальный объект **Global**.
5. Выделите в Структуре объекта элемент ***ApSource_Main*** и в свойстве Порт истории укажите 5388. Так же укажите стандартные настройки в соответствии с изображением.

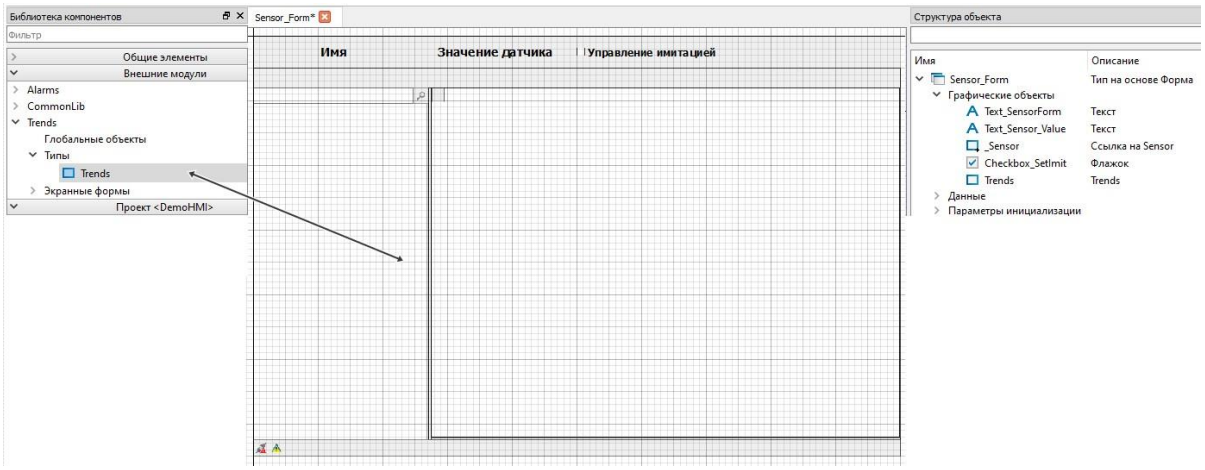


6. Сохраните изменения.

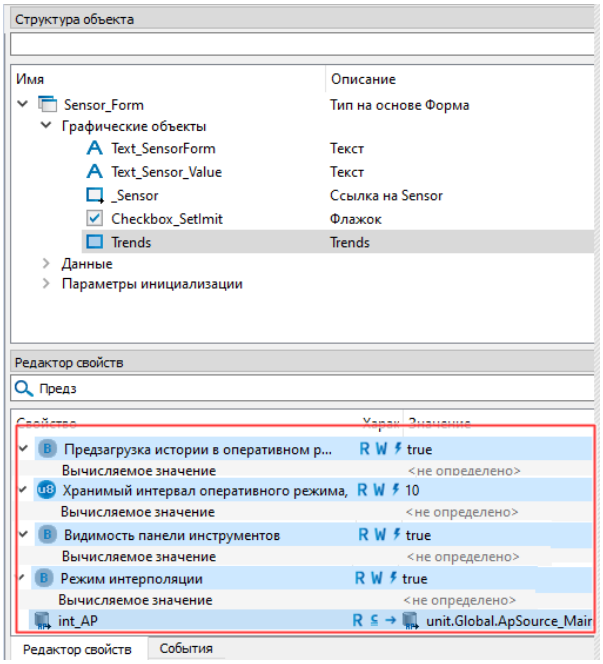
Добавление графиков в проект SePlatform.HMI

Модернизируем форму управления датчиком **Sensor_Form**.

1. Перейдите в форму **Sensor_Form**. Задайте ей ширину 1000, высота 650.
2. Из Библиотеки компонентов (вкладка Внешние модули) перетяните тип **Trends** на форму. В Структуре объекта назовите его Trends. Разместите все имеющиеся на форме элементы в поле зрения.

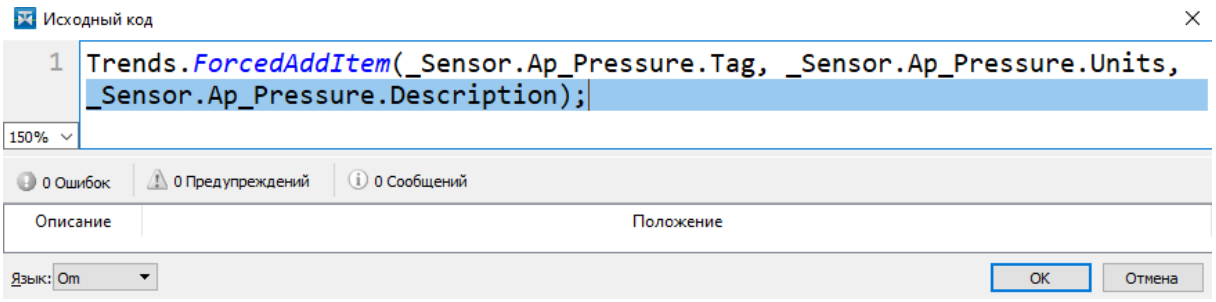


3. В Структуре объекта выделите элемент ***Trends***, для подключения трендов к источнику в свойстве int_AP сошлитесь на главный источник ApSource_Main, размещённый в глобальных объектах. Задайте остальные свойства в соответствии с изображением.



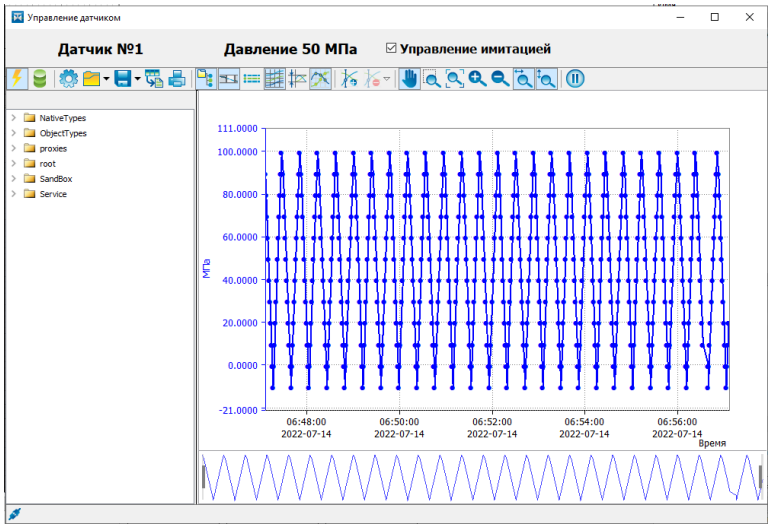
Необходимо, чтобы при открытии формы управления какого-либо датчика, сразу открывался соответствующий график. Для осуществления данной задачи нужно воспользоваться методом ForcedAddItem (добавляет сигнал в список отображаемых сигналов и график сигнала на трендовое поле), которая на вход принимает 3 строковых аргумента: путь до тега, единицы измерения и описание. Этот метод будем использовать при срабатывании события Opened у всей экранной формы.

4. В структуре объекта выделите форму ***Sensor_Form***, перейдите во вкладку События, кликните ПКМ по событию Opened → Добавить обработчик → Выполнить код. Введите скрипт в соответствии с изображением.



Теперь при открытии формы сразу должен открываться нужный график.

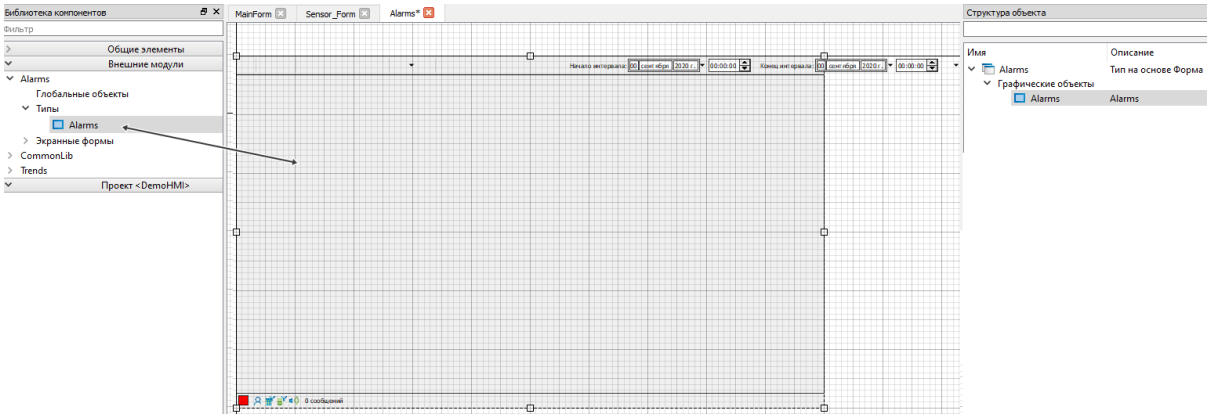
5. Сохраните проект, запустите его в RunTime. Откройте форму управления любого из датчиков. Теперь вместе с этой формой появляется соответствующий график.



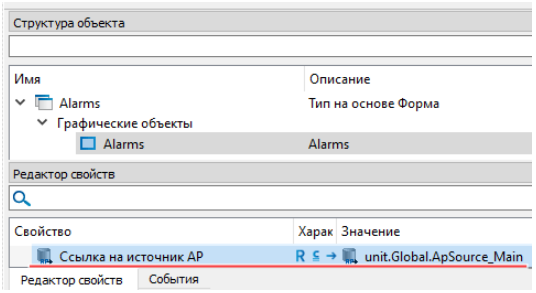
Отображение событий в проекте SePlatform.HMI

Теперь добавим ещё одну форму для отображения событий. Она будет универсальной и для датчиков, и для задвижек. Событий тоже будем фильтровать: при открытии формы любого из элементов, отображаются отфильтрованные события по соответствующему объекту.

1. Создайте новую экранную форму, назовите её Alarms_Form.
2. Перейдите в форму **Alarms_Form**, в свойстве Заголовок окна введите События, в свойстве Размеры окна: Автоподбор, Положение окна: По центру монитора. Установите ширину формы 1000, высоту – 600.
3. Из Библиотеки компонентов (вкладка Внешние модули) перетяните тип **Alarms**. В Структуре объекта назовите его Alarms.



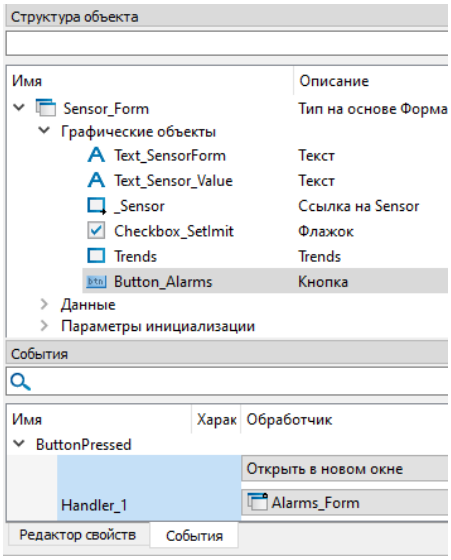
4. Для того, чтобы ***Alarms*** получали данные от источника, в свойстве Ссылка на источник AP сошлитесь на главный источник ApSource_Main, который находится в глобальном объекте **Global**.



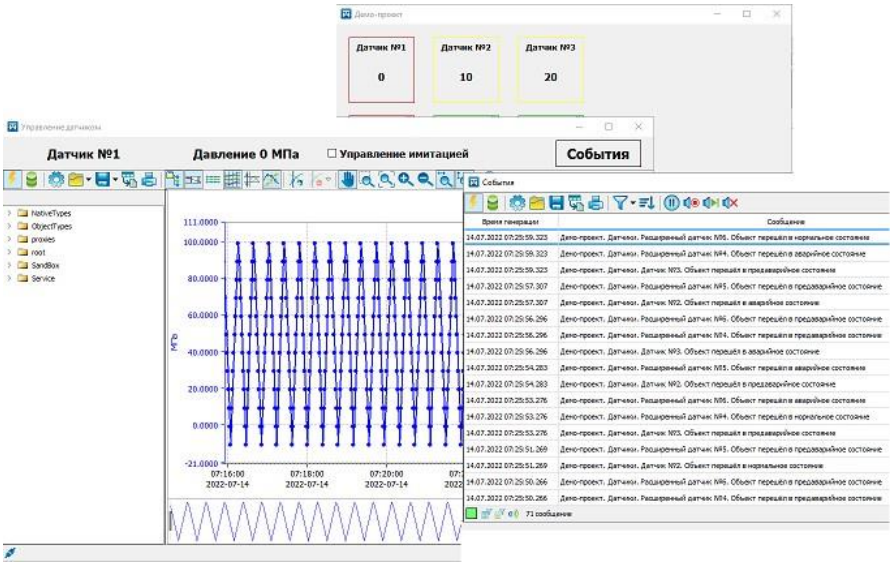
Теперь нужно здесь осуществить процесс фильтрации. Для того, чтобы это сделать, будем использовать метод SetAdvancedFilter (фильтрует отображаемые сообщения о событиях по указанному столбцу таблицы сообщений.) Нужно фильтровать по столбцу Источник (идентификатор данного столбца – source).

5. Перейдите на форму **Sensor_Form**. Создайте здесь ***Кнопку*** с именем Button_Alarms (при нажатии на эту кнопку будет открываться форма Alarms_Form). В свойстве Текст введите События. Настройте размер шрифта.

6. В Структуре объекта выделите Кнопку ***Button_Alarms***, перейдите во вкладку События, щёлкните ПКМ по ButtonPressed → Добавить обработчик → Открыть в новом окне → Alarms_Form.



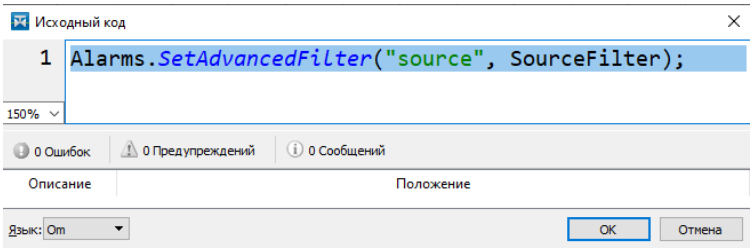
7. Сохраните проект, запустите в RunTime. Откройте форму управления одного из датчиков, нажмите на кнопку События. Убедитесь, что события (пока ещё со всего проекта) отображаются.



Теперь настроим саму фильтрацию.

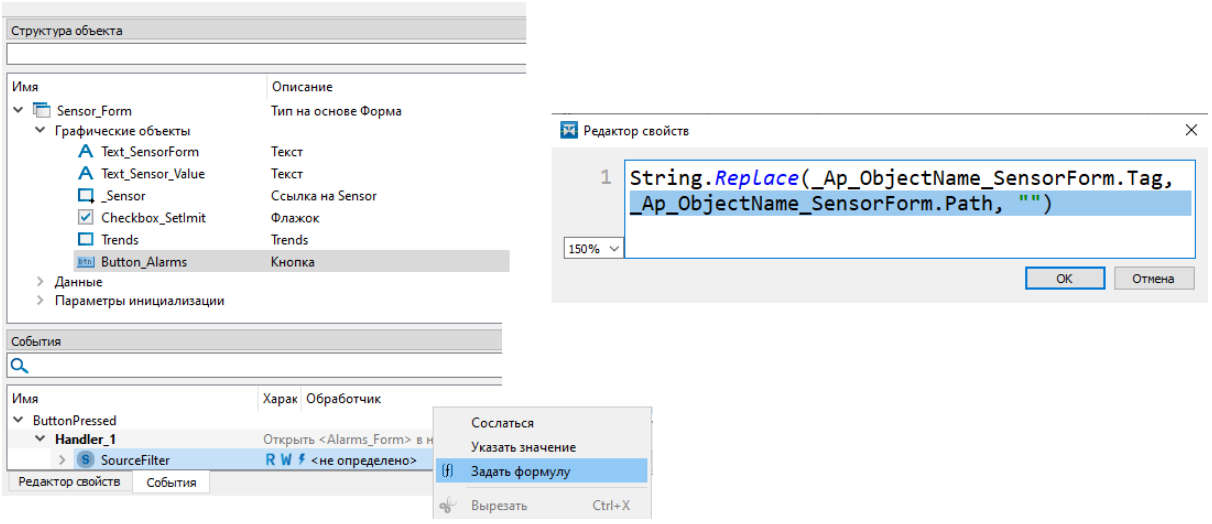
8. Перейдите в форму **Alarms_Form**, создайте здесь ***Уведомляющее поле*** → уведомляющий string. В Структуре объекта назовите его SourceFiltre (здесь от родительского окна к дочернему будет передаваться путь).

9. В Структуре объекта выделите форму ***Alarms_Form***, перейдите во вкладку События, щёлкните ПКМ по событию Opened → Добавить обработчик → Выполнить код → Редактировать. Введите код в соответствии с изображением.

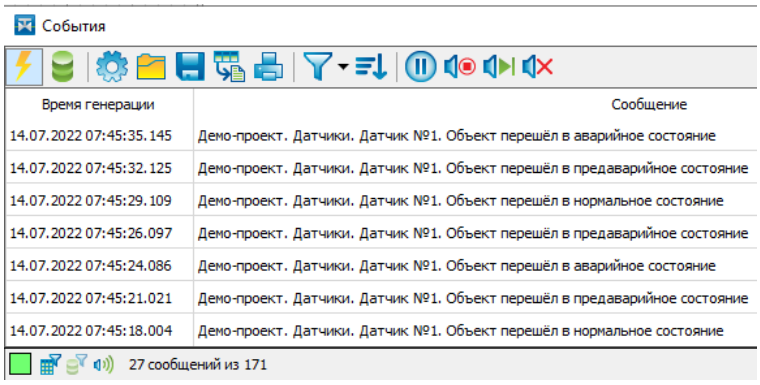


Теперь нужно в это ***Уведомляющее поле*** при открытии окна передать значение.

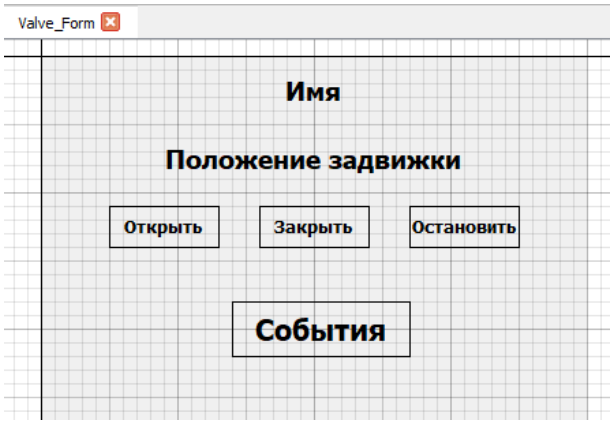
10. Перейдите в форму **Sensor_Form**, в Структуре объекта выделите Кнопку ***Button_Alarms***, перейдите во вкладку События, раскройте список события ButtonPressed, раскройте список обработчика Handler, найдите свойство SourceFiltre, щёлкните по нему ПКМ → Задать формулу. Введите формулу в соответствии с изображением (то есть берём тег с именем формы, удаляем из него путь, который дописывали и меняем его на пустоту).

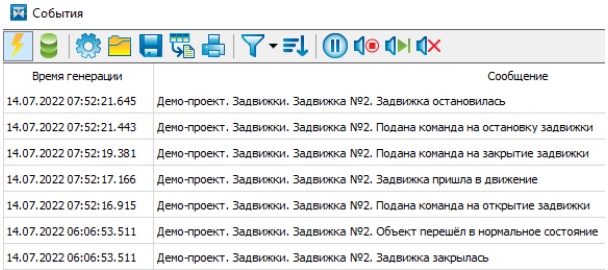


11.Сохраните проект, запустите его в RunTime. Откройте форму управления любого из датчиков, нажмите на кнопку События. Теперь на форме отображаются события по конкретному датчику.



Самостоятельное задание. Добавьте кнопку на форме управления задвижкой, которая будет открывать форму с отфильтрованными событиями (воспользуйтесь методом String.Replace).



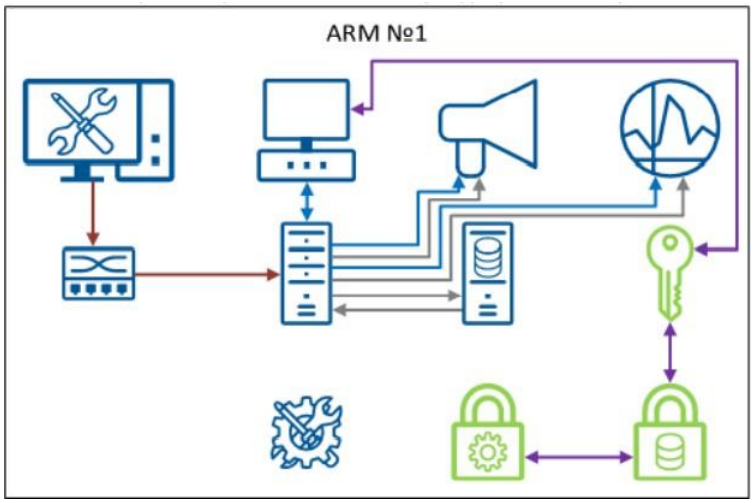


Сохраните проект, передайте его на машину с ОС Linux, проверьте работу.

Использование безопасности. Установка SePlatformSecurity. Конфигурирование OpenLDAP

В проектах автоматизации на различных производствах и не только очень важно обеспечить безопасность контролируемого технологического процесса, обеспечить аутентификацию пользователей для разграничения прав и предотвратить несанкционированный доступ. За обеспечение безопасности в

проекте отвечает компонент *SePlatform.Security*, который состоит службы из **SePlatform.Security.Agent**, отвечающей за аутентификацию и распределение прав пользователей, и **OpenLDAP** являющийся защищённой базой данных пользователей и их прав. Также есть приложение *SePlatform.Security Конфигуратор*, с помощью которого осуществляется конфигурирование OpenLDAP.



Прежде чем устанавливать дистрибутивы давайте закроем *SePlatform.HMI* на обеих машинах предварительно сохранив проект. Мы будем загружать в *SePlatform.HMI* дополнительные библиотеки, и для того, чтобы они были правильно загружены и установлены, необходимо закрыть программу.

Так как разработка проекта кроссплатформенная, в этом разделе будем делать акцент преимущественно на ОС Linux (всю подробную информация по установке *SePlatform.Security* можно найти в документации).

1. Откройте PuTTY, убедитесь в том, что Вы находитесь в папке с установочными файлами, введите команду для установки *SePlatform.Security*: **sudo dpkg –i SePlatform.security********.deb**.
2. Для того, чтобы настроить OpenLDAPвведите команду **sudo apt-get install slapd ldap-utils**. После вопроса «Хотите продолжить?» введите Y (yes). Откроется окно настройки пакета.

OpenLDAP – это хранилище, в хранилище могут находиться несколько баз данных безопасности, здесь Вы указываете пароль для хранилища, то есть пароль для того, чтобы подключиться. Обычно по умолчанию используется пароль **secret**.

3. В поле Пароль администратора введите **secret**, нажмите ОК.
4. Повторно введите пароль **secret**.
5. Чтобы указать свои данные в конфигурации OpenLDAP, переконфигурируйте его командой **sudo dpkg-reconfigure slapd**. Ответьте на следующие вопросы в соответствии с изображением (в полях для пароля вводите **secret**).

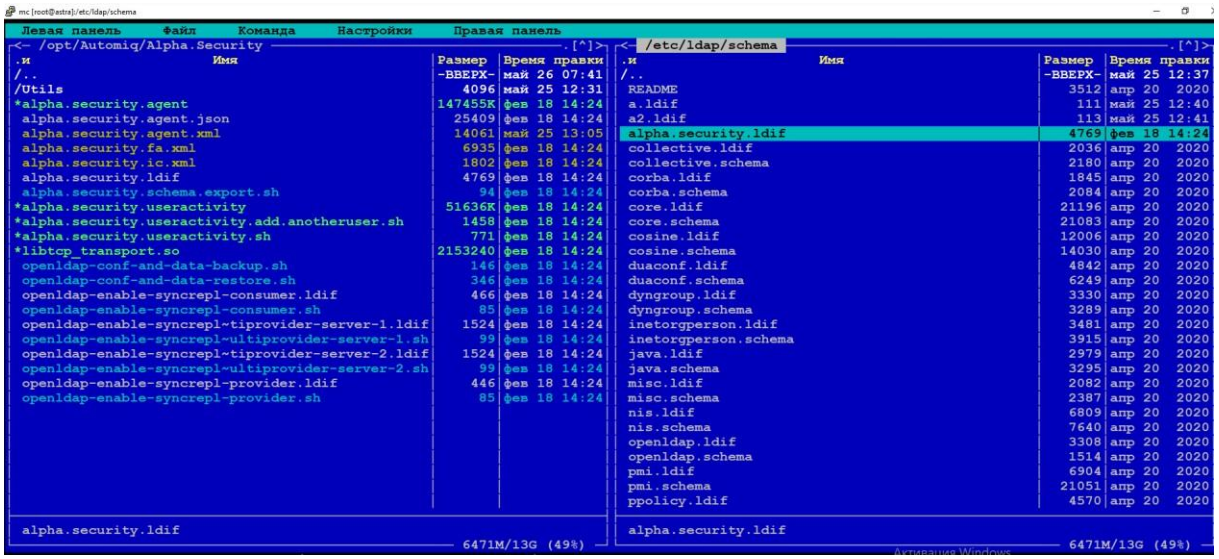
- Не выполнять настройку сервера OpenLDAP? - **«Нет»**
- Доменное имя DNS - **«maxcrc.com»**
- Название организации - <название> (можно указать любое)
- Пароль администратора - <пароль>
- Повторный пароль - <пароль>
- Используемые серверы баз данных - **«HDB»**
- Удалять базу данных при вычитке slapd? - **«Да»**
- Переместит старую базу данных? - **«Да»**, если есть файлы

6. Перейдите в Midnight Commander путём ввода команды **sudo mc**. Перейдите в папку /opt/SePlatform/SePlatform.Security, выделите файл **SePlatform.security.ldif**.

7. Для того, чтобы перейти в правую часть коммандера нажмите клавишу Tab.

8. В правой части коммандера перейдите в папку /etc/ldap/schema. Вернитесь в левую часть коммандера нажатием клавиши Tab.

9. Скопируйте из /opt/SePlatform/SePlatform.Security в /etc/ldap/schema файл **SePlatform.security.ldif** нажатием на клавишу F5.



10. Вернитесь к командной строке, нажав на клавишу F10. Следующие команды необходимо вводить, находясь в режиме суперпользователя. Для перехода в этот режим введите команду **sudo su** (теперь для ввода команд не нужно в начале использовать слово sudo).

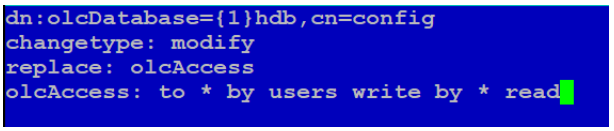
11. Перейдите в директорию /etc/ldap/schema введя команду **cd /etc/ldap/schema**.

12. Находясь в директории /etc/ldap/schema, выполните команду **ls *.ldif | xargs -t -n 1 ldapadd -Q -Y EXTERNAL -H ldapi:/// -f**.

13. Находясь в директории /etc/ldap/schema, выполните команду **ldapadd -Q -Y EXTERNAL -H ldapi:/// -f SePlatform.security.ldif**.

14. Перейдите в Midnight Commander, введя команду **mc**. Перейдите в /etc/ldap/schema. Нажмите комбинацию клавиш Shift+F4 для создания файла. Пропишите в нём строчки, описанные ниже (для вставки текста используйте ПКМ или комбинацию клавиш Shift+Insert):

```
dn:olcDatabase={1}hdb,cn=config
changetype: modify
replace: olcAccess
olcAccess: to * by users write by * read
```



15. Нажмите клавишу F2 для сохранения. Здесь же назовите этот файл ***a.ldif***. Выйдите из файла нажатием клавиши F10. Выйдите в командную строку (F10).

16. Проверьте, что Вы находитесь в директории /etc/ldap/schema. Находясь в этой директории введите команду **sudo ldapmodify -Y EXTERNAL -H ldapi:// -W -f a.ldif**. Введите тот самый пароль, который был указан при установке: **secret** (ввод будет зашифрован, поэтому никаких символов при вводе Вы не увидите).

17. Перейдите в Midnight Commander, введя команду **mc**. Перейдите в /etc/ldap/schema. Нажмите комбинацию клавиш Shift+F4 для создания файла. Пропишите в нём строчки, описанные ниже (для вставки текста используйте ПКМ или комбинацию клавиш Shift+Insert):

dn:olcDatabase={1}hdb,cn=config

changetype: modify

replace: olcAccess

olcAccess: {0}to * by users write by * read

```
dn:olcDatabase={1}hdb,cn=config
changetype: modify
replace: olcAccess
olcAccess: {0}to * by users write by * read
```

18.Нажмите клавишу F2 для сохранения. Здесь же назовите этот файл ***a2.ldif***. Выйдите из файла нажатием клавиши F10. Выйдите в командную строку (F10).

19.Проверьте, что Вы находитесь в директории /etc/ldap/schema. Находясь в этой директории введите команду **sudo ldapadd -Y EXTERNAL -H ldapi:// -W -f a2.ldif**. Введите тот самый пароль, который был указан при установке: **secret** (ввод будет зашифрован, поэтому никаких символов при вводе Вы не увидите).

OpenLDAP на ОС Linux переконфигурирован.

Так же установим на машину с ОС Linux пакет для *SePlatform.HMI*, чтобы он мог работать с библиотекой безопасности.

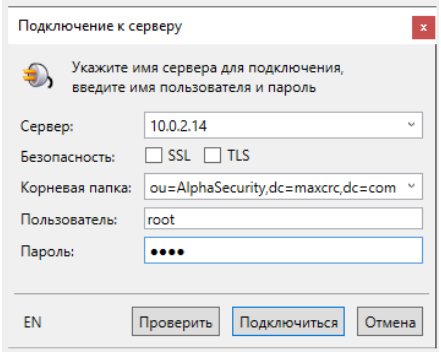
20.Откройте PuTTY, введите в командную строку команду **exit** для выхода из режима суперпользователя.

21.Убедитесь, что Вы находитесь в папке с установочными файлами и введите команду установки: **sudo dpkg –i SePlatform.hmi.security********.deb**

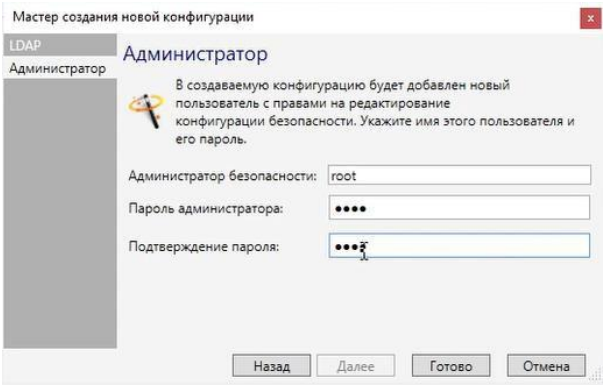
Теперь необходимо установить конфигуратор безопасности. Чтобы это сделать на машине с ОС Windows, установите дистрибутив: *SePlatform.Security*. Этот компонент устанавливает сразу несколько компонентов: *SePlatform.Security* – это SePlatform.Security агент, который нужен на каждой локальной машине для того, чтобы *SePlatform.HMI* умел работать с компонентами безопасности. OpenLDAP уже установлен на
линуксе, компонент Security для HMI – компонент, который нужен, чтобы *SePlatform.HMI* умел взаимодействовать с безопасностью. Он подложит специальную библиотеку в папку с установленным *SePlatform.HMI*. *Конфигуратор Security* – это тот самый конфигуратор, который нужен для того, чтобы сконфигурировать базу, то есть добавить пользователей и приложение.

Работа с SecurityConfigurator

1. Запустите установленный *Конфигуратор* из меню Пуск → SePlatform → SecurityConfigurator. В строке Сервер введите IP-адрес машины с ОС Linux, уберите галочку с TLS. Придумайте имя
пользователя и пароль для него. Нажмите Подключиться.

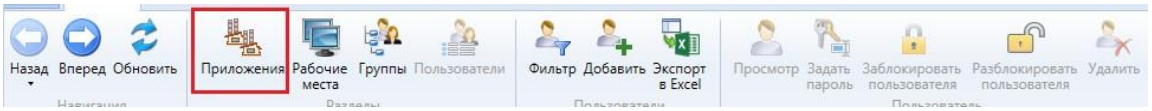


2. «Создать ли новую конфигурацию?» → Да.
3. В строке Администратор LDAP меняем «Manager» на «Admin». Пароль администратора LDAP – **secret**. Нажмите Далее.
4. Вводите имя пользователя, которого хотите создавать и пароль для этого пользователя. После этого нажмите Готово.

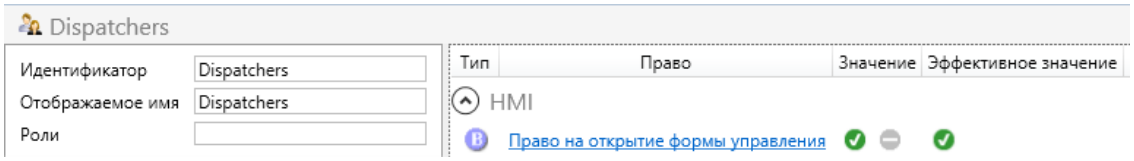


Начнём создавать пользователей. Важное замечание: пользователи должны состоять в группе, так как права распространяются на группу, а не на конкретного пользователя.

5. Нажмите на кнопку Приложения.

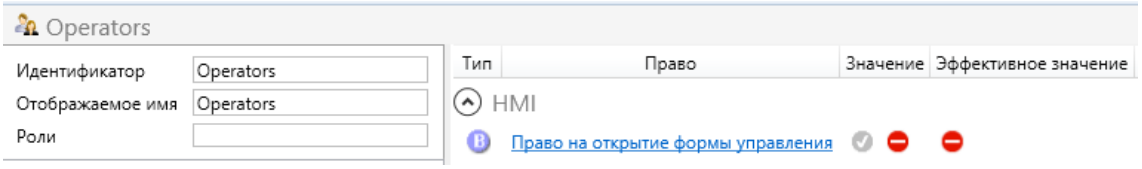


6. Нажмите на кнопку Добавить и введите название HMI.
7. Для того, чтобы создать право, нажмите на кнопку Логическое право. Назовите его OpenPopup (право на открытие формы).
8. Чтобы дать ему более понятное описание, сначала сохраните изменения и перейдите внутрь этого права. В свойстве Описание введите: Право на открытие формы управления. Сохраните изменения и снова перейдите в приложение.
9. Нажмите на кнопку Группы → Добавить. Назовите группы Dispatchers.
10. Нажмите на кнопку Править и здесь – Добавить права. Выберите право OpenPopup внутри HMI. Явно укажите, что это право разрешено для группы Диспетчеров.



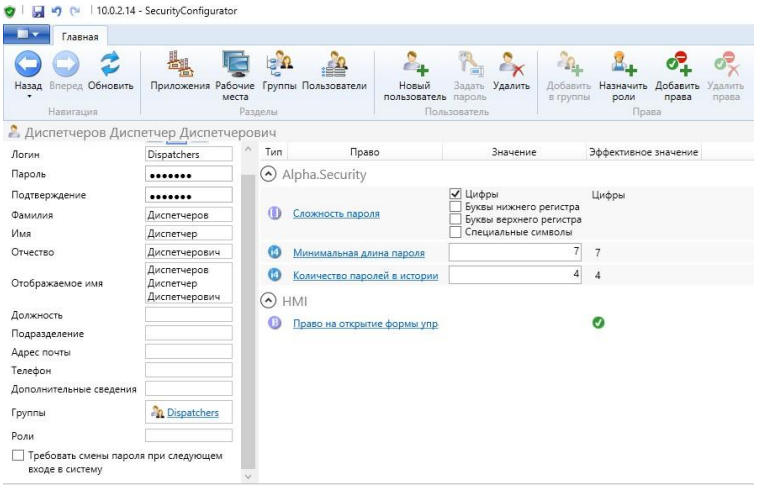
11. Сохраните изменения.

Самостоятельное задание. Создайте группу Операторов, запретите им право на открытие формы управления.



Теперь можем переходить к созданию пользователей.

12. Нажмите на кнопку Пользователи. Введите логин, пароль, состоящий из 7 символов и ФИО пользователя.
13. Нажмите на кнопку Добавить в группы → Dispatchers. Снимите галочку с Требовать смены пароля при следующем входе в систему. Сохраните изменения.



Самостоятельное задание. Создайте нового пользователя, добавьте его в группу операторов.

Настройка SePlatform.secutiry.agent.XML

Настроим **SePlatform.security.agent** на машине с ОС Windows.

1. Перейдите в папу C:\Program Files\SePlatform\SePlatform.Security, откройте **SePlatform.secutiry.agent.XML** с помощью NotePad++.
2. В строке с тэгом **EntryPointNetAgent** введите адрес той машины, на которой Вы находитесь – IP-адрес Wndowsмашины.

```
7       -->
8       <EntryPointNetAgent Address="10.0.2.17" Port="1010"/>
9
```

3. В строке с тэгом **LDAPServer** введите адрес машины, на которой сконфигурирован LDAPServer – IP-адрес Linux машины.

```
15      -->
16      <LdapHosts>
17          <LDAPServer Address="10.0.2.14" Port="389"/>
18          <!-- <LDAPServer Address="199.99.99.111" Port="389"/> -->
19      </LdapHosts>
```

Если OpenLDAP сконфигурирован на ОС Windows то в строке с тэгом LdapUser в атрибуте value указать cn=manager, если же на ОС Linux, то cn=admin.

4. В строке с тэгом **LdapPassword** необходимо ввести хэшированный пароль. Захешировать его можно с помощью специальной программы SePlatform.security.crypter.exe через командную строку:
    1) Откройте через меню Пуск командную строку.
    2) Для перехода в папку с программой введите **cd C:\Program Files\SePlatform\SePlatform.Security\Utils** нажмите Enter.
    3) Для запуска программы введите **SePlatform.security.crypter.exe**.
    4) Введите пароль secret и скопируйте получившуюся строку (для сохранности вставьте в отдельный блокнот). Вставьте этот пароль в строку с тэгом **LdapPassword**.

```
24      <!-- Пароль LDAP -->
25      <LdapPassword value="qPQg3EuTdhKym8Cp9AQzTPHeaCWG6X/gGXbRwKcHKhVeEjW4cVHbXQYBHf3XH8SUc
26
```

5. Отключите выполнение контроля целостности: в строке с тэгом **Options** в атрибуте **ICMode** введите значение 0.

```
153
154     <Options LoggerLevel="2" ICMode="0" kbDriverString="0x1D+0x38+0x53;0x1D+0x2A+0x01;" UseRightsCacheStorage="0" />
155
```

6. Сохраните **SePlatform.security.agent.XML**.
7. Откройте Диспетчер задач → Службы, перезапустите службу **SePlatform.Security.Agent**.

Теперь настроим **SePlatform.security.agent** на машине с ОС Linux.

8. Откройте PuTTY, перейдите в Midnight Commander при помощи команды **sudo mc**. Перейдите в директорию /opt/SePlatform/SePlatform.Security, откройте файл **SePlatform.security.agent.hml** нажатием клавиши F4.
9. В строке с тэгом **EntryPointNetAgent** введите адрес той машины, на которой Вы находитесь – IP-адрес Linux машины.

```
------>-->
------><EntryPointNetAgent Address="10.0.2.14" Port="1010"/>
```

10. В строке с тэгом **LDAPServer** введите адрес машины, на которой сконфигурирован LDAPServer – IP-адрес Linux машины.

```
><LdapHosts>
><------><LDAPServer Address="10.0.2.14" Port="389"/>
><------><!--<LDAPServer Address="192.168.56.1" Port="636"/>-->
></LdapHosts>
```

Если OpenLDAP сконфигурирован на ОС Windows то в строке с тэгом LdapUser в атрибуте value указать cn=manager, если же на ОС Linux, то cn=admin.

11. В строку с тэгом **LdapPassword** необходимо ввести хэшированный пароль, который был сгенерирован в пункте 4 (пароль **secret**).

```
><!-- Пароль LDAP -->
><LdapPassword value="gPQg3EuTdhKym8Cp9AQzTPHeaCWG6X/gGXbRwKcHKhVeEjW4cVHbXQYBHf3XH8SUdyDo7XmSSy4i1eIKyD6KHq9EF45huBF13+25bqUhVuu81UK/LD
```

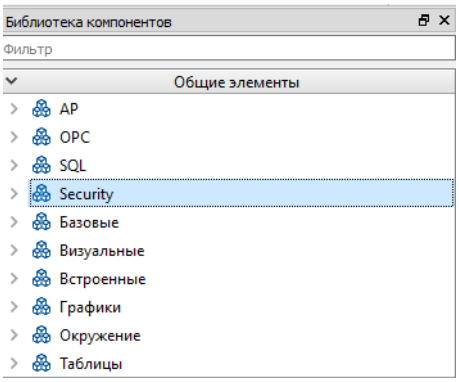
12. Отключите выполнение контроля целостности: в строке с тэгом **Options** в атрибуте **ICMode** введите значение 0.

```
><Options LoggerLevel="2" ICMode="0" UseRightsCacheStorage="0" FAMode="1"/>
```

13. Сохраните **SePlatform.security.agent.hml** нажатием клавиши F2, перейдите в командную строку нажатием клавиши F10.
14. Перезапустите службу командой **sudo systemctl restart SePlatform.Security**.
15. Проверьте её статус: **sudo systemctl status SePlatform.Security**.

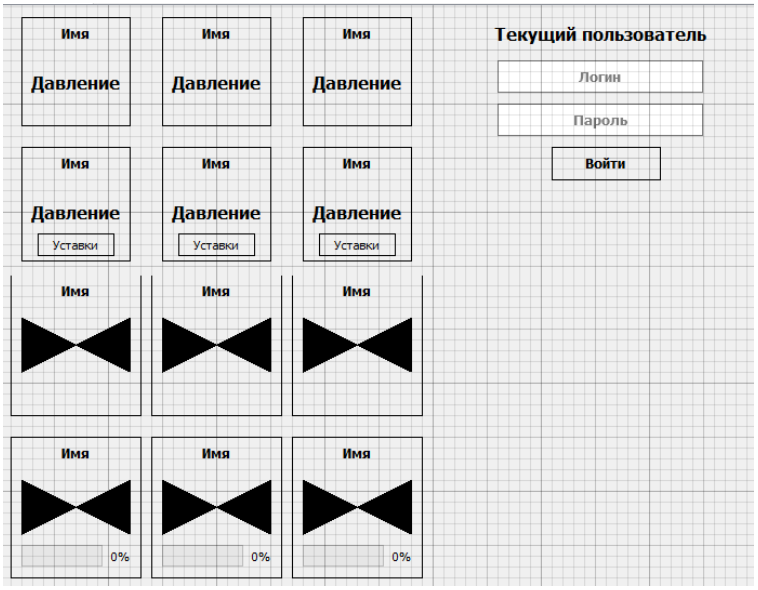
Добавление в проект компонентов безопасности

Откройте *SePlatform.HMI* на машине с ОС Windows, откройте проект. Обратите внимание, в библиотеке компонентов добавился раздел Security.



На главной форме создадим поле для аутентификации. Разместим на главной форме текстовое поле, куда будет выводиться логин текущего пользователя, добавим 2 поля ввода: логин и пароль. Также добавим кнопку, нажимая на которую будет производиться вход.

1. На форме **MainForm** разместите элемент ***Текст***, назовите его Text_CurrentUser. В свойстве Текст укажите Текущий пользователь. Настройте шрифт и положение текста.
2. Добавьте сюда же 2 ***Поля ввода***: TextEdit_UserName и TextEdit_Password. В Свойстве Текст заполнителя укажите Логин и Пароль соответственно. Настройте шрифт и положение текста. У поля ввода ***TextEdit_Password*** в свойстве Скрывать ввод укажите значение true для того, чтобы никто не увидел пароль пользователя.
3. Из Библиотеки компонентов перетяните на форму **MainForm** элемент ***Кнопка***, назовите её Button_Login. В свойстве Текс укажите Войти. Настройте шрифт и положение текста.

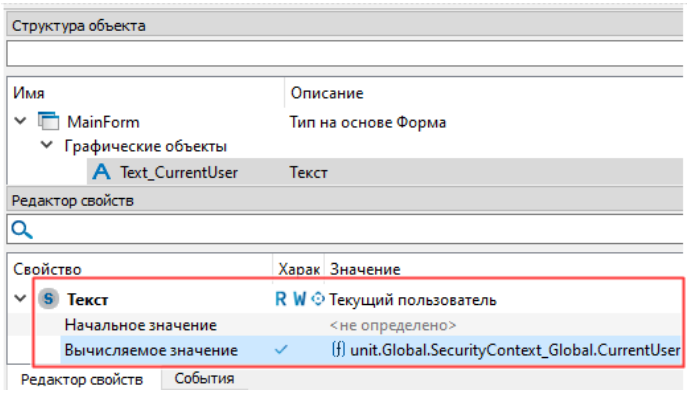


Для того, чтобы взаимодействовать с безопасностью, необходимо использовать элементы безопасности. Элементы безопасности мы поместим в глобальные объекты, т.к. они будут использоваться всюду.

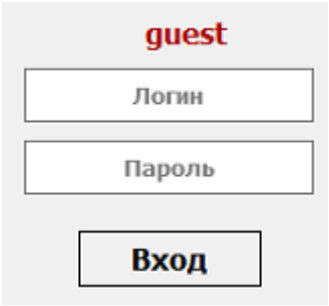
4. Перейдите в глобальный объект **Global**, из Библиотеки компонентов (раздел Security) перетяните сюда элемент ***Контекст безопасности***. В Структуре объекта назовите его SecurityContext_Global.

Добавление ***Контекста безопасности*** в глобальные объекты означает то, что в проекте будет взаимодействие с агентом безопасности.

5. Перейдите в **MainForm**. Для того, чтобы вывести текущего пользователя, выделите в Структуре объекта ***Text_CurrentUser***, перейдите в Вычисляемое значение его свойства Текст, введите сюда скрипт в соответствии с изображением:

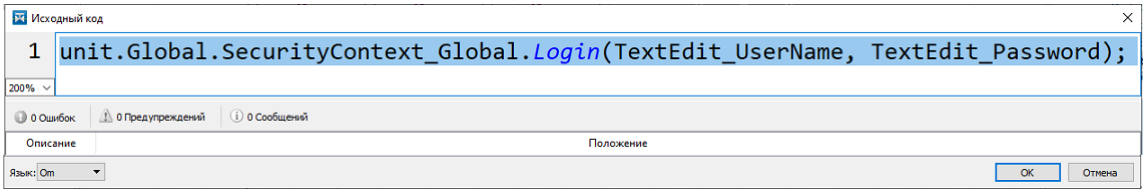


6. Сохраните проект, запустите в RunTime.

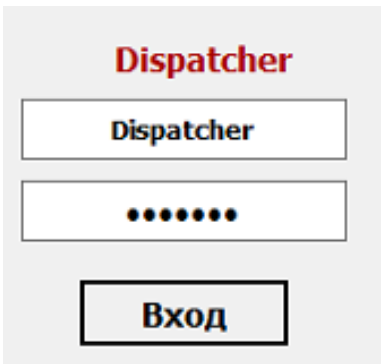


Теперь начнём обеспечивать вход конкретному пользователю.

7. Выделите ***Кнопку Button_Login***, перейдите во вкладку События, щелкните ПКМ по событию ButtonPressed → Выполнить код → Редактировать. Введите скрипт в соответствии с изображением.



8. Сохраните проект, запустите в RunTime. Попробуйте зайти под Оператором и под Диспетчером.

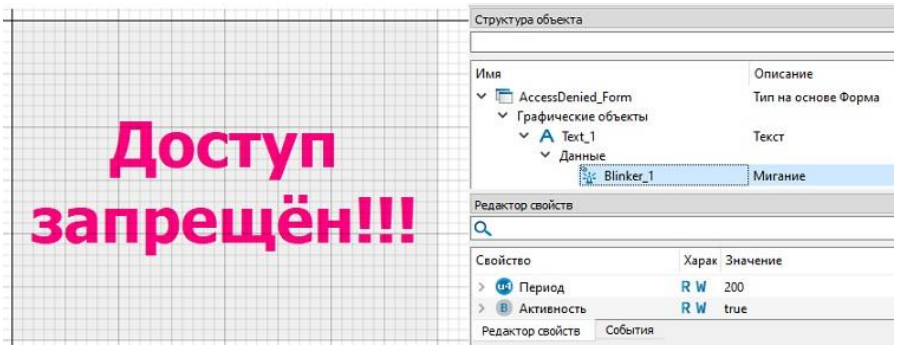


Когда пользователю будет разрешено право OpenPopup, то он сможет открыть форму управления, если запрещено, то нужно вызвать форму с текстом «Доступ запрещён».

9. Создайте новую экранную форму, назовите её AccessDeied_Form. Зайдите внутрь настройте свойства Размеры окна и Положение окна.

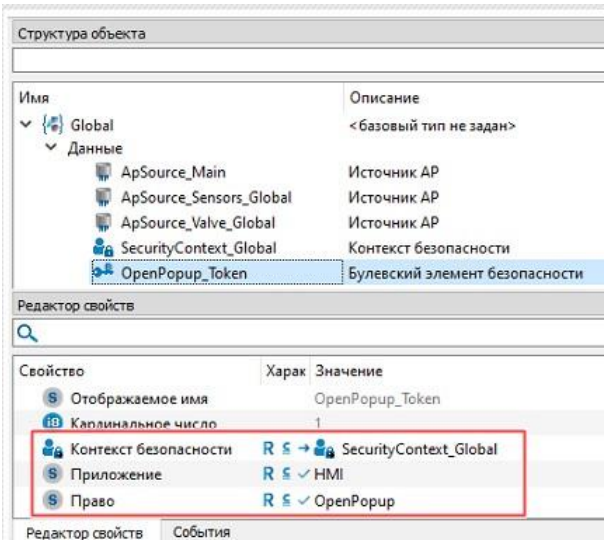
10.Перетяните на форму элемент ***Текст*** с надписью Доступ запрещён!!!. Настройте шрифт и положение текста.

11.Перетяните на ***Текст*** «Доступ запрещён!!!» элемент ***Мигание*** (раздел Визуальные). В свойстве Период укажите 200, Активность true.



Для того, чтобы явно указать, что в проекте будет взаимодействовать с безопасностью (конкретнее с булевским правом), необходимо добавим ***Булевский элемент безопасности***.

12.Перейдите в глобальный объект **Global** и перетяните сюда из Библиотеки компонентов (раздел Security) ***Булевский элемент безопасности***, назовите его OpenPopup_Token. Укажите свойства в соответствии с изображением (значения свойств Приложение и Право берутся из SecurityConfigurator).



Сейчас форма управления открывается сразу без каких-либо проверок, как только происходит событие MouseClick.

Теперь нужно, чтобы при срабатывании события MouseClick сначала происходила проверка, а потом, если она пройдена, открывалось дополнительное окно. Если она не пройдена, то должно открываться окно с сообщением «Доступ запрещён!!!».

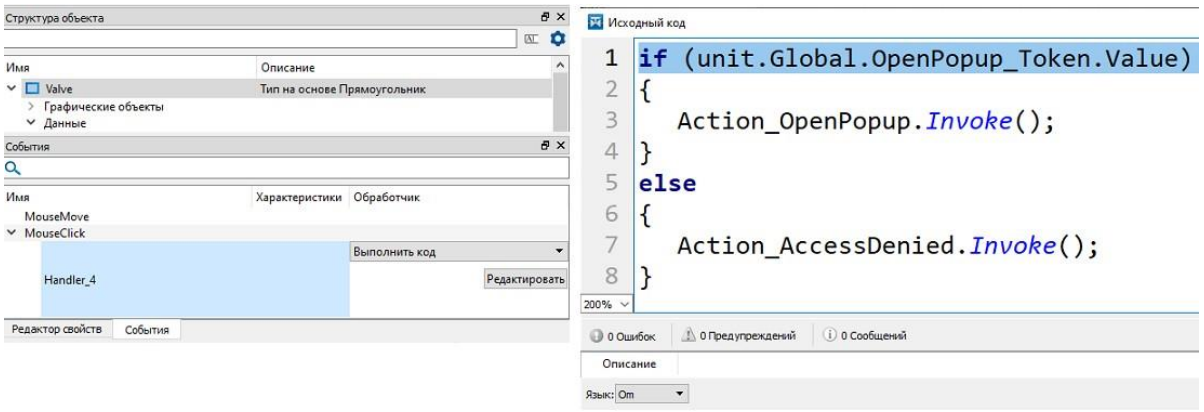
Для того, чтобы из обработчика вызвать другие обработчики, которые будут зависеть от каких-то условий, необходимо создать дополнительные обработчики в форме. Для этого используется встроенный компонент, который называется ***Команда***.

13.Перейдите к типу **Valve**. Из Библиотеки компонентов (раздел Встроенные) перетяните сюда элемент ***Команда***. Назовите его Action_AccessDenied (Эта команда будет вызвана, если проверка не будет пройдена).

14.Выделите в Структуре объекта Команду ***Action_AccessDenied***, перейдите во вкладку События, щелкните ПКМ по событию Invoked → Добавить обработчик → Открыть в диалоговом окне → AccessDenied_Form.

15.Из Библиотеки компонентов (раздел Встроенные) перетяните сюда элемент ***Команда***. Назовите его Action_OpenPopup (эта команда будет вызвана в случае успешной проверки). Здесь необходимо настроить обработчик открытия нового окна. Чтобы снова все ссылки не прокидывать выделите в Структуре объекта прямоугольник ***Valve***, перейдите во вкладку События, раскройте список события MouseClick, скопируйте имеющийся обработчик и вставьте его в событие Invoked команды ***Action_OpenPopup***.

16.Выделите в Структуре объекта прямоугольник ***Valve***, перейдите во вкладку События, удалите имеющийся обработчик в событии MouseClick. Добавьте новый обработчик → Выполнить код. Введите код в соответствии с изображением.



17.Сохраните проект, запустите его в RunTime. Залогиньтесь под Диспетчером, при нажатии на задвижку, открывается форма управления. Залогиньтесь под Оператором. При нажатии на задвижку открывается форма **AccessDenied_Form**.

Самостоятельное задание. Необходимо осуществить проверку доступа для Датчиков. Диспетчер может открывать форму управления датчиками, Оператор – нет.

10. Резервирование

Зачастую на разных уровнях автоматизации резервируют различные системы. Резервирование по сути повышает надёжность проекта. Проект должен быть безопасным, надёжным и предсказуемым в том числе и с точки зрения мониторинга и управления технологическим процессом.

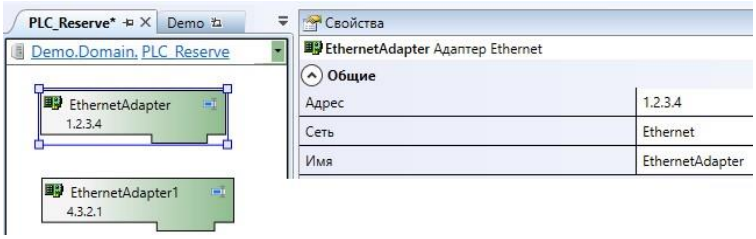
Для осуществления резервирования с точки зрения контроллера (то есть когда на каком-то объекте стоит контроллер, и, чтобы в случае выхода из строя этого контроллера продолжалась автоматизация управления технологическим процессом) обычно на предприятиях ставят резервные контроллеры, которые также работают с этим же оборудованием и могут общаться между собой, чтобы понимать, у кого на данный момент будет управление. То есть они могут быть в режиме полного дублирования функционала, либо в режиме резервирования, когда они вместе работают с одним и тем же набором оборудования, но при этом распределяют роли между собой.

Резервирование источников

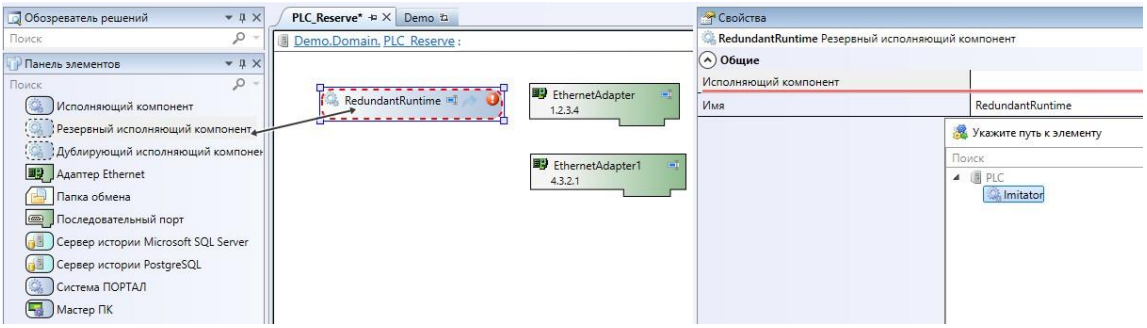
В этом случае необходимо добавить ещё один компонент, который будет резервировать тот самый вычислитель, с которым по умолчанию работает сервер.

1. Откройте *DeveloperStudio*. Через Обозреватель решений перейдите в пространство ***Domain***.
2. Из Панели элементов перетяните в ***Domain*** элемент ***Компьютер***, назовите его PLC_Reserve.
3. Перейдите внутрь ***PLC_Reserve***, выделите ***EthernetAdapter***, в его свойстве Адрес введите IP (для примера укажем IP-адрес несуществующей машины). В свойстве Сеть укажите Ethernet.

Можно добавить для примера ещё один ***EthernetAdapter*** с другим адресом.



4. Для того, чтобы сервер понимал, что это новое устройство не просто так существует, а нужно для резервирования внутри ***PLC_Reserve*** разместите ***Резервный исполняющий компонент*** (из Панели элементов). В свойстве Исполняющий компонент укажите, что он будет резервировать Imitator, находящийся в PLC.



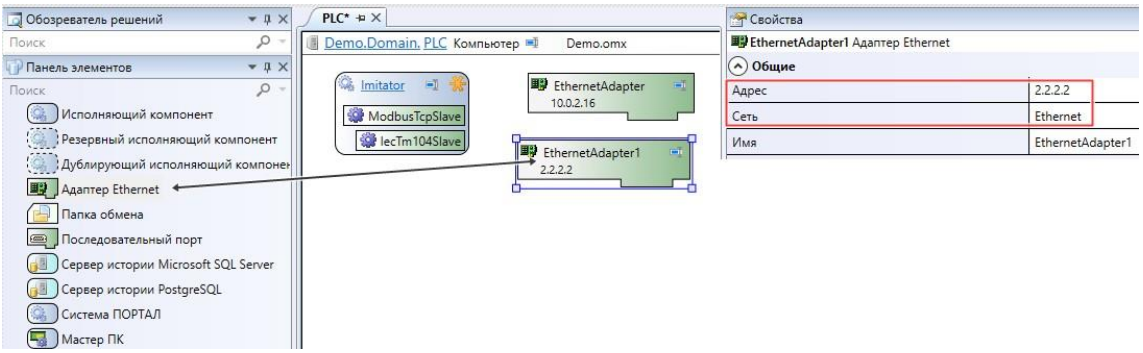
5. **Постройте решение**, **перейдите к Мастеру развёртывания** и **примените конфигурацию** к линуксовому серверу.

Вот и всё, резервирование контроллеров настроено. То есть достаточно указать, где находится резервное устройство и какую программу оно выполняет. В случае, если произойдёт разрыв с источником, сервер должен знать, что есть второй точно такой же. А переключение между контроллерами программируется внутри самих контроллеров.

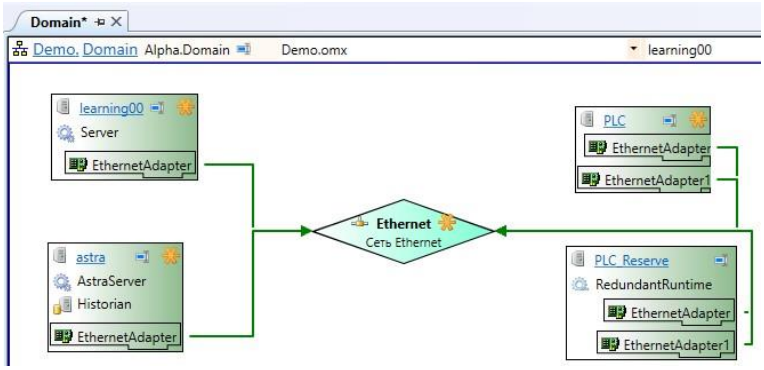
Посмотреть информацию о том, с каким контроллером идёт обмен, можно посмотреть через *OpcExplorer* в ветке сервисных сигналов Service.

Также у станции могут быть резервные каналы связи.

6. Для добавления резервного канала связи перейдите в ***Компьютер PLC*** при помощи Обозревателя решений.
7. Внутрь ***Компьютера PLC*** из Панели элементов перетяните ещё один ***EthernetAdapter***, в свойстве Адрес укажите IP-адрес несуществующей машина (в качестве примера), в свойстве Сеть укажите Ethernet.



Теперь при построении конфигурации у источника будет добавлен ещё один канал, ну и в случае разрыва связи по одному из каналов, связь может быть восстановлена по другому каналу.



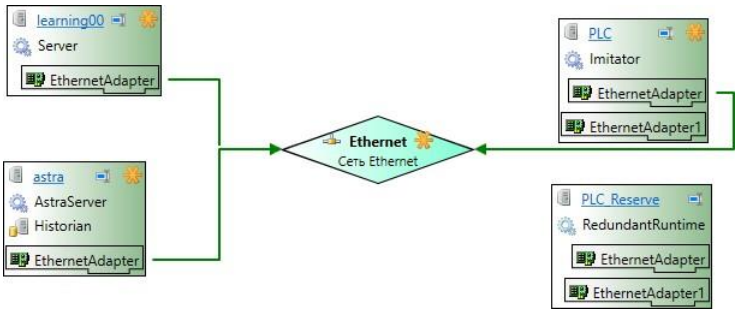
8. **Постройте решение**, **перейдите к Мастеру развёртывания** и **примените конфигурацию** к линуксовому серверу.

Резервирование серверов

В *SePlatformServer* поддерживается режим горячего резервирования. Это значит, что у Вас в RunTime поднято 2 сервера с одинаковой конфигурацией, то есть они уже исполняются, они уже запущены, и не нужно тратить время на инициализацию каких-то резервных махинаций. И в случае, если необходимо переключиться, серверу достаточно просто переключить состояние. В этом суть горячего резервирования. То есть исполняющий компонент уже работает. Не нужно будет тратить время в случае разрыва соединения, выхода из строя или отключения питания. Резервный сервер сразу готов подхватить работу.

Для организации резервирования серверов необходимо *SePlatformServer* развернуть на разных машинах. В этом и есть суть резервирования. Вы резервируете полноценные среды исполнения. Соответственно резервная пара не может быть развёрнута внутри одной машины. И в качестве резервной машины мы будем использовать машину с ОС Windows.

Перед осуществлением резервирования серверов удалите связи с несуществующими ***EthernetAdapter***.



1. Перейдите в ***Узел learning00*** (виндовая машина). Удалите имеющийся здесь сервер.
2. Перетяните сюда из Панели элементов ***Резервный SePlatform.Server***. В свойстве Резервируемый сервер необходимо указать, какой сервер он будет резервировать (AstraServer). В разделе свойств Основной канал внутри Адаптер основного укажите EthernetAdapter Linux машины, Адаптер резервного – EthernetAdapter Wndowsмашины.

[скриншот из DevStudio]

Обратите внимание, у этого резервного сервера теперь одна и та же конфигурация с ***AstraServer***, тот же самый набор модулей

Чтобы резервирование работало, был заранее отключён брэндмауэр и все ***Узлы SePlatform.Domain*** названы в соответствии с именами машин.

Резервирование Systeme Platform-серверов настроено. Теперь необходимо настроить развёртывание, чтобы
SePlatform.Domain мог развернуть конфигурацию резервного сервера.

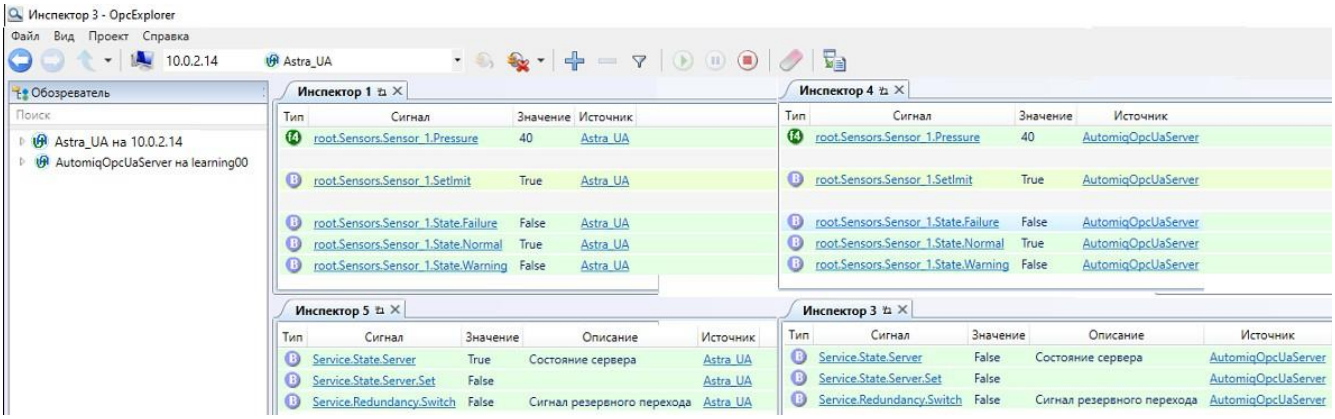
3. Перейдите в **SePlatform.domain.agent.XML** (C:\Program Files\SePlatform\SePlatform.Domain). В строке с тэгом **Component** необходимо ввести имя исполняющего компонента, который находится внутри узла Wndowsмашина: RedundantServer.

```
43          -->
44          <Component InstalledName="Server_1" Name="RedundantServer" StorageLimitSize="0" StorageLimitNum="0"/>
45      </Components>
```

4. Сохраните **SePlatform.domain.agent.XML**, перезапустите службы **SePlatform.Net. Agent** и **SePlatform.Domain.Agent** через Диспетчер задач.

5. Вернитесь в *DeveloperStudio*. **Постройте решение**, **перейдите к Мастеру развёртывания** и **примените конфигурацию** на обеих машинах.

Для просмотра результата можно воспользоваться *OpcExplorer*. Подключитесь к виндовому и линуксовому OPC UA server, добавьте 4 окна Инспектор (вкладка Проект → Добавить инспектор). Разместите в Инспекторах информацию в соответствии с изображением (в левой части OpcExplorer данные по линуксовому источник, справа – по виндовому).



В режиме резервирования резервные сервера могут быть в двух состояниях: в работе или в резерве. Сервер, который находится в работе, опрашивает поле, отправляет команды, генерирует события и отправляет историю. Сервер, который находится в резерве, поле не опрашивает, команды не отправляет, не генерирует события и историю не сохраняет. Его задача – это от основного сервера через канал репликации данных (через который они взаимодействуют между собой) получать актуальные данные. Соответственно, у вас есть рабочий сервер, который производит всю работу и есть резервный, который на подхвате. И в случае резервного перехода у резервного сервера уже будут актуальные данные, потому что он их получал с основного сервера. И после того, как произошел резервный переход, резервный сервер уже будет иметь актуальные данные, ему не нужно будет тратить время ни на загрузку, ни на общий опрос. Он просто будет продолжать работу. И буквально доли секунды нужно для того, чтобы всё это сработало.

За состояние сервера для того, чтобы определить, какой именно находится в работе, а какой в резерве, отвечает сигнал Service.State.Server. А за осуществление резервного перехода отвечает сигнал Service.Redundancy.Switch. Этот сигнал можно отправлять с любого сервера, не важно, в работе он или в резерве.

Для проверки попробуйте отследить значение давление датчика при отправлении сигнала резервного перехода или при осуществлении перезапуска рабочего сервера.

11. Работа с SePlatform.AccessPoint

SePlatform.AccessPoint выполняет роль конечной точки доступа к оперативным данным и событиям множества источников данных. Это точка доступа, которая консолидирует данные с разных машин. Вы к нему подключаетесь, и он сразу предоставляет данные с того сервера, который находится в работе. То есть через *AccessPoint* вы видите актуальные данные и можете отправлять управляющие воздействия. Он сам будет определять, с какими серверами нужно работать, будет сам передавать и собирать данные с рабочей машины. Вы к нему подключаетесь и видите уже актуальную информацию. Для того, чтобы использовать *AccessPoint*, нам нужно его сначала сконфигурировать.

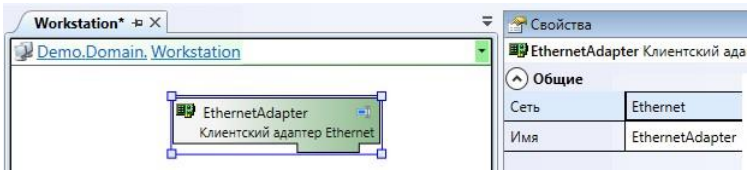
1. Откройте *DeveloperStudio*, при помощи Обозревателя решений перейдите в пространство ***Domain***. Из Панели элементов перетяните сюда ***Рабочее место*** (WorkStation). Элемент ***Рабочее место*** характеризует рабочие АРМы.

Вы можете один раз настроить *DeveloperStudio* и разворачивать конфигурацию сколько угодно раз.

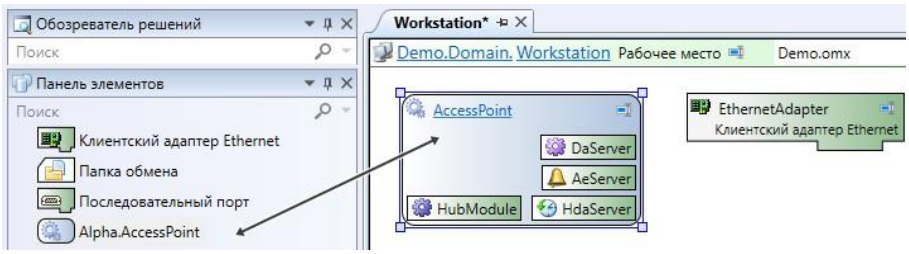
2. Перейдите внутрь ***WorkStation***.

Обратите внимание, здесь есть клиентский ***EthernetAdapter***, он отличается от ***EthernetAdapter***, который был в ***узлах*** и в ***компьютерах*** тем, что у него нет IP адреса. Это клиентская машина, это она цепляется к источникам, кто к ней будет цепляться, ей неважно. Учитывая то, что рабочее место может быть сконфигурировано один раз, а развёрнуто раз 10, IP-адрес не указывается. Здесь указывается только принадлежность к сети. *Access.Point* и все сервера, с которых он собирает данные, должны находиться в одной сети.

Выберите в свойстве сеть ***клиентского EthernetAdapter*** – Ethernet.

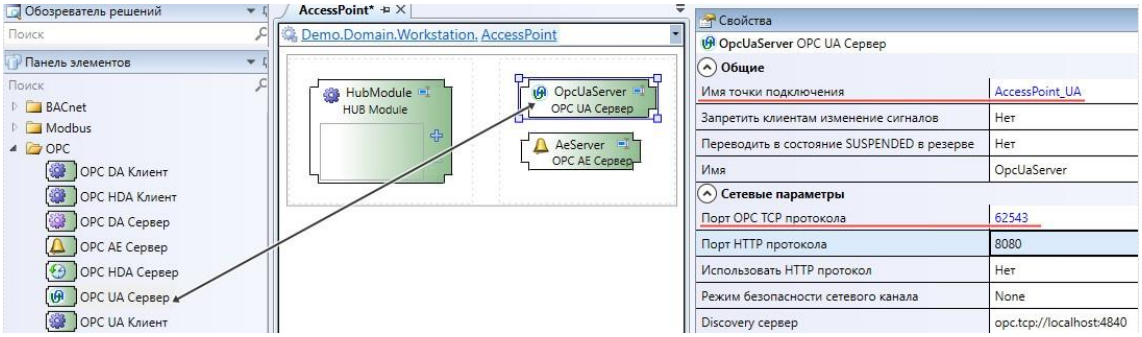


3. Внутрь ***WorkStation*** перетяните из Панели элементов ***SePlatformAccessPoint***.

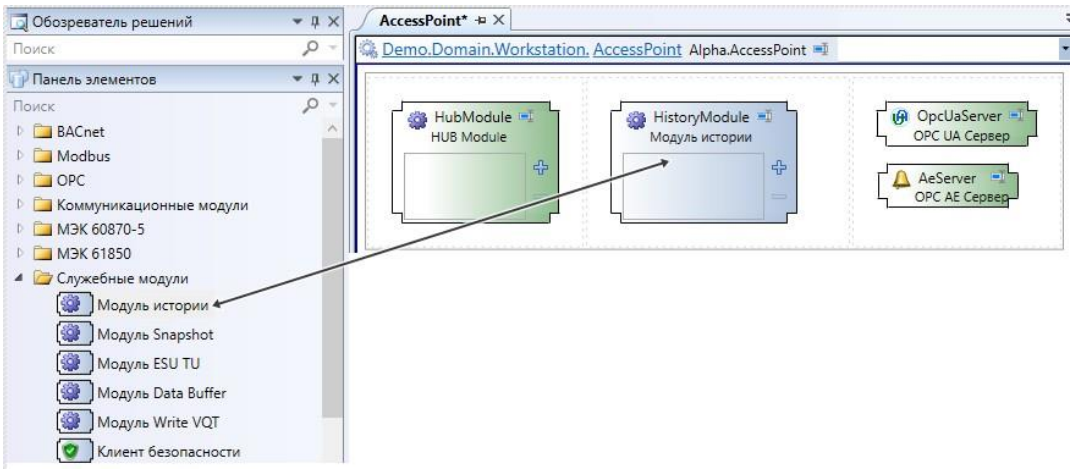


AccessPoint не может быть развернут на центральном узле, поэтому мы будем его разворачивать на машине с ОС Linux (в нашем случае она дочерняя).

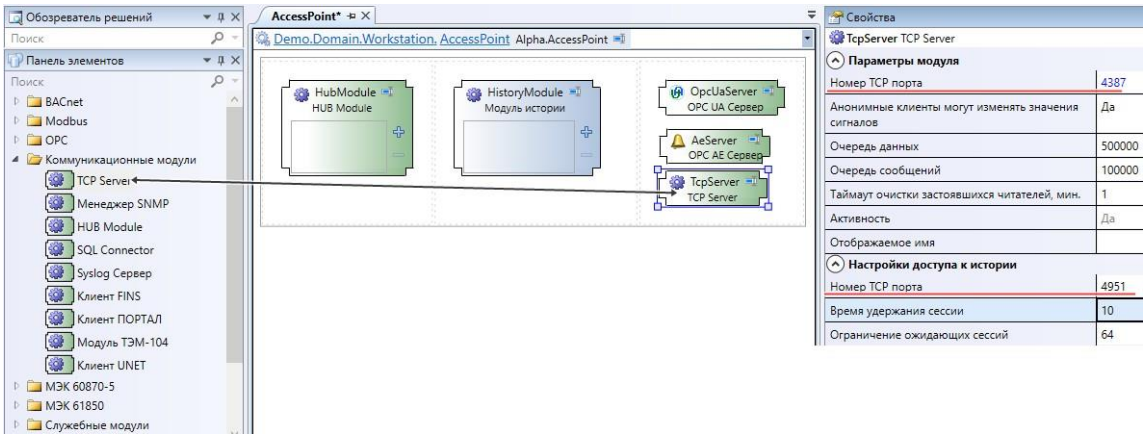
4. Перейдите внутрь ***AccessPoint***. Здесь уже добавлены модули DA, AE, HDA. Удалите модули DA и HDA (на ОС Linux они не работают). Из Панели элементов (раздел OPC) перетяните ***OPC UA Сервер***. В его свойстве Имя точки подключения введите AccessPoint_UA. В свойстве Порт OPC TCP протокола введите 62543 (чтобы не было конфликтов с OPC UA, расположенным в AstraServer).



5. Внутрь *AccessPoint* перетяните из Панели элементов ***Модуль истории*** (раздел Служебные модули) для того, чтобы *AccessPoint* мог работать с историей, но базы добавлять не нужно. О базах, с которыми нужно будет связываться, *AccessPoint* узнает после подключения.

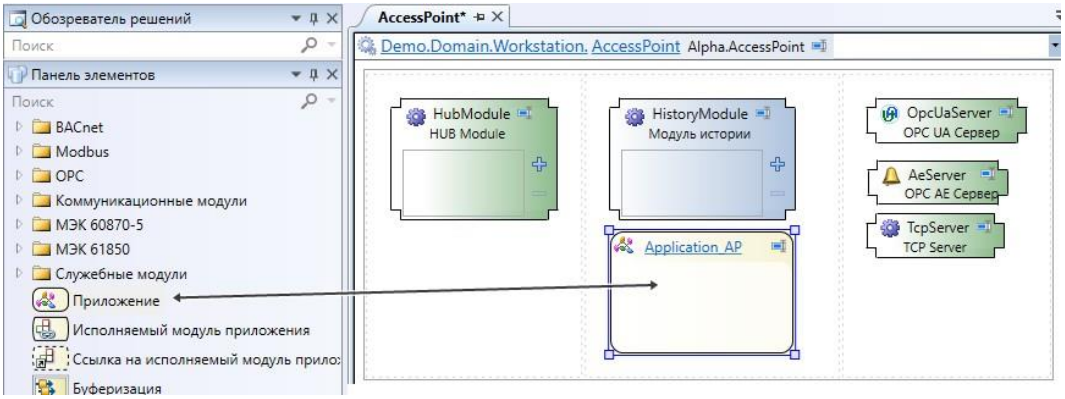


6. Для связи *AccessPoint* с *SePlatform.HMI* нужно добавить ***коммуникационный модуль TCP-сервер*** из Панели элементов, в его свойствах укажите номер TCP-порта: 4387 для оперативных данных, и номер TCP порта в настройках доступа к истории: 4951.

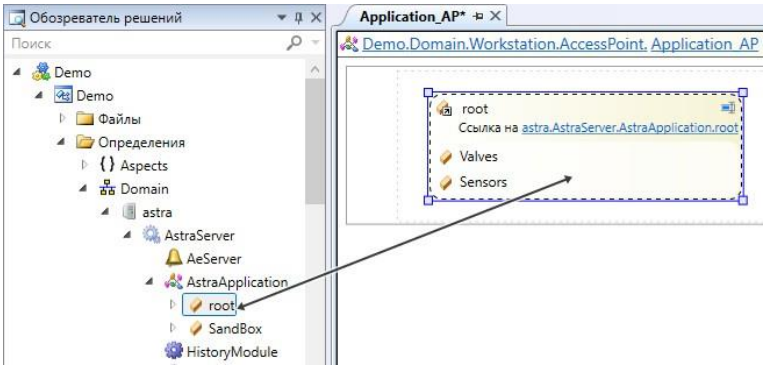


С коммуникационной точки зрения модули настроены. Теперь необходимо настроить то, какие данные будет транслировать *AccessPoint*.

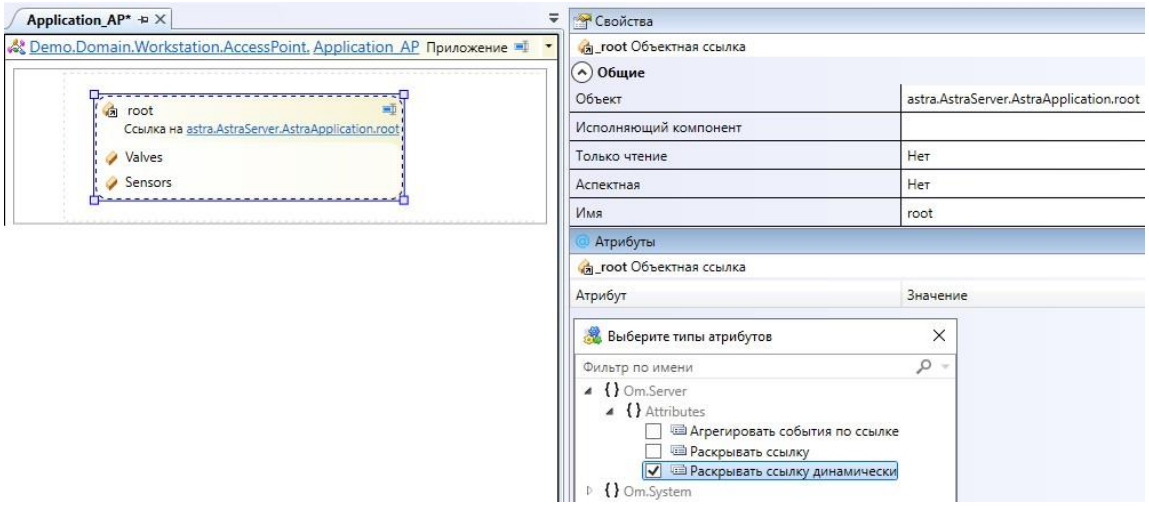
7. Внутрь ***AccessPoint*** из Панели элементов перетяните ***Приложение***. Назовите его Application_AP.



8. Перейдите внутрь ***Application_AP***. Здесь нужно указать, какая ветка будет транслироваться. Перетяните сюда из Обозревателя решений объект ***root***, который находится в ***AstraServer*** (как ссылку). Удалите в названии нижнее подчёркивание для того, чтобы в *SePlatform.HMI* не менять пути у источников и элементов АР.



9. Добавьте для объекта ***root*** внутри ***Application_AP*** атрибут Om/Server → Attributes → Раскрывать ссылку динамически. Этот атрибут используется для того, чтобы динамически транслировать дерево сигналов, которое есть у источника.



AccessPoint настроен, осталось настроить развёртывание.

10. Откройте **SePlatform.domain.agent.XML** (C:\Program Files\SePlatform\SePlatform.Domain). После закрытия тэга **Server** вставьте строки, описанные ниже:

```
<Domain>
    <WorkstationRoles>
        <WorkstationRole Name="Workstation">
            <KnownWorkstations>
                <KnownWorkstation SePlatformNetPath="ChildNode" Description="Точка доступа на астралинуксовой машине"/>
            </KnownWorkstations>
        </WorkstationRole>
    </WorkstationRoles>
    <ConfigurationCache Path="C:/temp"/>
</Domain>
```

11. Сохраните **SePlatform.domain.agent.XML**.
12. Через Диспетчер задач перезапустите службы **SePlatform.Net.Agent** и **SePlatform.Domain.Agent**, посмотрите информацию в журнале *EventLogViewer*.
13. Откройте PuTTY. Для установки *SePlatform.AccessPoint* убедитесь, что Вы находитесь в папке с установочными файлами и введите команду **sudo dpkg –i SePlatform.accesspoint ********.deb**.
14. Перейдите в Midnight Commander при помощи команды **sudo mc**, откройте **SePlatform.domain.agent.XML** нажатием клавиши F4 (/opt/SePlatform/SePlatform.Domain).

15.Для того, чтобы привязаться к службе, после строки с описанием службы на ОС Linux введите строку, описанную ниже:

```
<SePlatform.Server Name="Server_Workstation" ServiceName="SePlatform-accesspoint.service" />
```

16.После закрытия тэга **Server** вставьте строки, описанные ниже:

```
<Workstation Dynamic="false">
    <WorkstationRoles>
        <WorkstationRole Name="Workstation">
            <Components StoragePath="/usr/local/DomainStorage/cache/server">
                <Component InstalledName="Server_Workstation"
Name="AccessPoint"/>
            </Components>
        </WorkstationRole>
    </WorkstationRoles>
</Workstation>
```

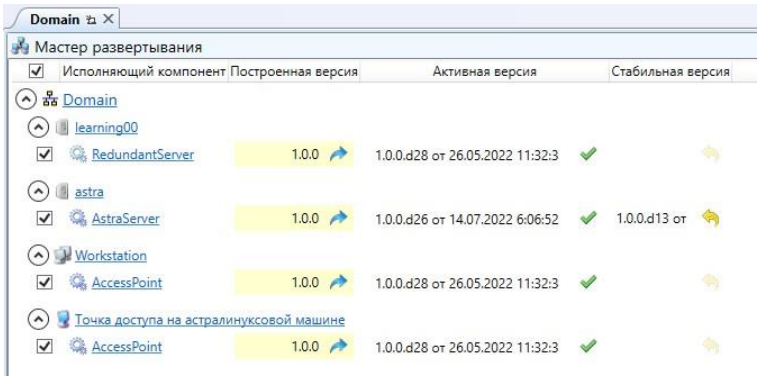
```
----><------></Components>
----></Server>

----><Workstation Dynamic="false">
----><WorkstationRoles>
---->    <WorkstationRole Name="Workstation">
----><------><Components StoragePath="/usr/local/DomainStorage/cache/server">
----><------>    <Component InstalledName="Server_Workstation" Name="AccessPoint"/>
----><------></Components>
---->    </WorkstationRole>
----></WorkstationRoles>
</Workstation>
```

17.Сохраните **SePlatform.domain.agent.XML** нажатием клавиши F2, перейдите к командной строке нажатием клавиши F10.

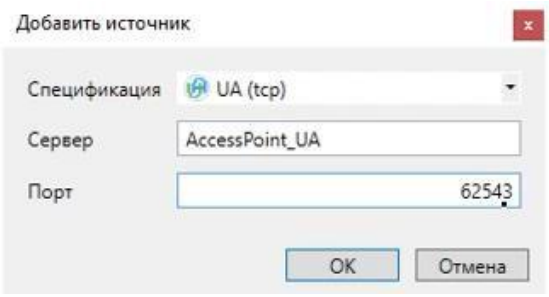
18. Перезапустите службы **SePlatform.Net.Agent** и **SePlatform.Domain.Agent**, посмотрите их статус:

sudo systemctl restart SePlatform.Net.service sudo systemctl status SePlatform.Net.service sudo systemctl restart SePlatform.Domain.service sudo systemctl status SePlatform.Domain.service

19. Откройте *DevStudio*, **постройте решение**, **перейдите к Мастеру развертывания**, **примените конфигурацию** на все имеющиеся машины.

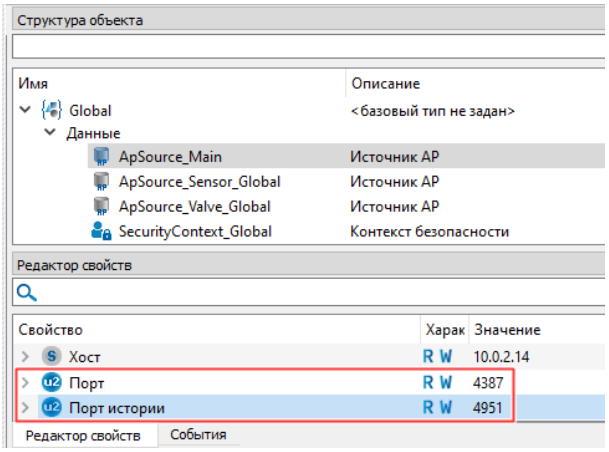


20. Посмотреть результат при переключении работы с одного сервера на другой можно в *OpcExplorer*. Подключитесь к OPC UA серверу AccessPoint_UA и добавьте ещё одно окно с инспектором для *AccessPoint*. Перетяните сюда те же сигналы, что и у виндового и линуксового серверов.



Также необходимо настроить *SePlatformHMI* так, чтобы он работал с *AccessPoint*.

21.Откройте проект в *SePlatform.HMI*. Перейдите в глобальный объект **Global**. Выделите в Структуре объекта главный источник ***ApSource_Main***. Добавьте ему свойства в соответствии с изображением.



22.Сохраните проект, запустите в RunTime. Всё как работало, так и работает, только теперь через *AccesPoint*.

12. Резервирование истории

1. Установите *SePlatform.Historian* на машине с ОС Windows.

Сервер всё пишет сразу в 2 базы. Соответственно, если что-то случилось с одной машиной, то на другой всегда есть актуальная история. Историю можно писать на ту локальную машину, где находится

сам сервер, так и на удалённые машины. В нашем случае история пишется на машину с ОС Linux, и мы также можем настроить SePlatformHistorian на машину с ОС Windows.

2. Откройте *DevStudio*, перейдите с помощью Обозревателя решений в ***Узел*** Wndows машины (learning00).

3. Перетяните сюда из Панели элементов ***Дублирующий SePlatform.Historian***. В Свойстве Дублируемый сервер выберите базу, которую он будет дублировать.

Для того, чтобы это всё заработало, необходимо указать то же имя этой базе данных внутри **SePlatform.Historian.Server.HML**.

4. Откройте на машине с ОС Windows **SePlatform.Historian.Server.HML** (внутри папки C:\Program Files\SePlatform\SePlatform.Historian). Для отключения DCOM в строку с тэгом **SePlatform.Historian.Server** необходимо дописать **EnableDCOM="0"**, тем самым отключив поддержку DCOM.

```
24  <Alpha.Historian.Server StatPort="3388" DefaultPrimaryDir="c:\Alpha.Historian\Databases" MaxPoolMemorySize="512" EnableDCOM="0">
```

5. Спуститесь ниже к тэгу **<Base>**. Перед закрытием данного тэга необходимо раскомментировать строчку для определения базы данных. Удалите комментарий и в атрибуте **Alias** введите название для данной базы: History_Database.

```
102         -->
103         <Base Alias="History_Database" PreferredCommonCacheLimit="128" ActiveStorageDepth="3" StorageDepth="365" />
104     </Bases>
```

6. Сохраните **SePlatform.Historian.Server.HML**, перезапустите службу **SePlatform.Historian.Server** через Диспетчер задач.

7. Откройте *DevStudio*, **постройте решение**, **перейдите к Мастеру развёртывания** и **примените конфигурацию** на все имеющиеся машины.

Теперь сервер пишет историю сразу в 2 базы.

Заключение

В данном учебном материале описан базовый функционал и порядок действий для создания первого простого проекта. Это лишь малая часть возможностей *SePlatform*. Более подробную информацию о функционале компонентов можно найти в документации. Стоит также учитывать, что компоненты *SePlatform* могут использоваться не только в проектах АСУ ТП. Для углубления знаний о компонентах *SePlatform* рекомендуются курсы:

- Systeme Platform. Расширенный курс разработки (в разработке)
- Systeme Platform. Курс администратора (по запросу)