

НАСТРОЙКА И РАБОТА REGUL OPC UA SERVER

Руководство пользователя

DPA-302.6

Версия ПО 1.7.1.0

Версия 1.6

Июль 2023

История изменений руководства пользователя

| Версия руководства пользователя | Описание изменения |
|---------------------------------|---|
| 1.2 | <p>Добавлена история изменений руководства пользователя.</p> <p>Добавлены знаки с предупреждающей и поясняющей информацией.</p> <p>Добавлены новые разделы:</p> <ul style="list-style-type: none"> – «Специальные структуры для простых переменных»; – «Ограничение анонимного доступа к серверу OPC UA»; – «Обращение в службу технической поддержки». <p>Раздел «Включение OPC UA сервера»: скорректировано описание настройки включения сервера.</p> <p>Раздел «Описание конфигуратора»: дополнено описание настройки параметров точек подключения.</p> <p>Раздел «Влияние запуска и останова приложений на работу сервера OPC UA»: добавлено описание генерируемых сервером событий.</p> <p>Дополнительно по тексту внесены небольшие изменения с уточняющей информацией</p> |
| 1.3 | <p>Раздел «Организация адресного пространства сервера»: добавлено описание опции, позволяющей сворачивать описание о типах объекта.</p> <p>Раздел «Элементарные типы данных»: сняты ограничения на размер строковых переменных OPC UA.</p> <p>Раздел «Описание конфигуратора. Поле «Настройки»»: добавлено описание о возможности включения/отключения событий OPC UA.</p> <p>Дополнительно по тексту внесены небольшие изменения с уточняющей информацией</p> |
| 1.4 | <p>Раздел «Передача различных типов данных с помощью протокола OPC UA»: добавлен новый подраздел – «Конвертация текста в переменных типа STRING». Добавлена информация о поддержке опции code_page для осуществления конвертации русского/английского текста в переменных типа STRING</p> |
| 1.5 | <p>Обновление в связи с выпуском среды разработки Astra.IDE.</p> <p>Добавлены новые разделы:</p> <ul style="list-style-type: none"> – «Отображение статических переменных в адресном пространстве UA сервера»; – «Приложение А. Настройка конфигурационного файла ServerConfig.xml». <p>Подраздел «Включение OPC UA сервера»: добавлено описание об изменении запускаемой по умолчанию версии OPC UA в зависимости от установленной версии СПО</p> <p>Раздел «Влияние запуска и останова приложений на работу сервера OPC UA» заменен на раздел «События, генерируемые OPC UA сервером» с дополнением и актуализацией содержимого</p> |
| 1.6 | <p>Внесены небольшие изменения с уточняющей информацией</p> |

АННОТАЦИЯ

Сервер REGUL OPC UA позволяет клиентам осуществлять доступ по чтению и записи к данным IEC-приложений, работающих на программируемых логических контроллерах серии REGUL. Настройка осуществляется с помощью программного обеспечения Astra.IDE.

Данное руководство предназначено для эксплуатационного персонала и инженеров-проектировщиков АСУ ТП которые должны:

- иметь, как минимум, среднее техническое образование;
- приступить к работе только после изучения данного руководства.


Обновление информации в Руководстве

Производитель ООО «РегЛаб» оставляет за собой право изменять информацию в настоящем Руководстве и обязуется публиковать более новые версии с внесенными изменениями. Обновленная версия Руководства доступна для скачивания на официальном сайте Производителя: <https://reglab.ru/>.


Для своевременного отслеживания выхода новой версии Руководства рекомендуется оформить подписку на обновление документа. Для этого необходимо на сайте Производителя: <https://reglab.ru/> кликнуть на кнопку «Подписаться на обновления» и оставить свои контактные данные.

В руководстве присутствуют знаки с предупреждающей и поясняющей информацией. Каждый знак обозначает следующее:

ПРЕДУПРЕЖДАЮЩИЕ ЗНАКИ

| | |
|---|--|
|  | <p>ВНИМАНИЕ! Здесь следует обратить внимание на способы и приемы, которые необходимо в точности выполнять во избежание ошибок при эксплуатации или настройке.</p> |
|---|--|

ИНФОРМАЦИОННЫЕ ЗНАКИ

| | |
|---|--|
|  | <p>ИНФОРМАЦИЯ Здесь следует обратить внимание на <u>важную</u> информацию</p> |
|---|--|

СОДЕРЖАНИЕ

| | |
|--|-----------|
| АННОТАЦИЯ | 3 |
| СОДЕРЖАНИЕ | 4 |
| Передача различных типов данных с помощью протокола OPC UA | 5 |
| Организация адресного пространства сервера | 5 |
| Конвертация текста в переменных типа STRING | 6 |
| Трансляция данных | 6 |
| Элементарные типы данных | 6 |
| Специальные структуры для простых переменных | 7 |
| Трансляция перечислимого типа | 11 |
| Отображение массивов структур в адресном пространстве UA сервера | 11 |
| Отображение статических переменных в адресном пространстве UA сервера .. | 12 |
| Конфигурирование сервера OPC UA | 14 |
| Подключение к ПЛК | 14 |
| Включение OPC UA сервера | 14 |
| Описание конфигурирования | 15 |
| Поле «Сертификаты» | 17 |
| Поле «Настройки» | 17 |
| Настройка соединений с OPC UA клиентами | 22 |
| Ограничение анонимного доступа к серверу OPC UA | 22 |
| Файлы и каталоги, используемые при работе компонента | 23 |
| Добавление переменных в адресное пространство сервера OPC UA | 24 |
| События, генерируемые OPC UA сервером | 28 |
| Устранение неполадок | 31 |
| Клиент не может установить соединение с сервером | 31 |
| Не отображаются пользовательские переменные | 31 |
| Обращение в службу технической поддержки | 32 |
| Приложение А Настройка конфигурационного файла ServerConfig.xml | 33 |

ПЕРЕДАЧА РАЗЛИЧНЫХ ТИПОВ ДАННЫХ С ПОМОЩЬЮ ПРОТОКОЛА OPC UA

Организация адресного пространства сервера

Сервер отображает в своем адресном пространстве как данные, так и типы данных. Например, в приложении, работающем на контроллере, имеются следующие типы данных:

```

TYPE composite_t :
  STRUCT
    m_i : INT;
    m_f : REAL;
    m_s : STRING;
  END_STRUCT
END_TYPE

TYPE super_composite_t :
  STRUCT
    m_sub1 : composite_t;
    m_q : INT;
    m_sub2 : composite_t;
    m_z : REAL;
    m_sub3 : composite_t;
  END_STRUCT
END_TYPE
    
```

и следующие переменные:

```

cmp : composite_t;
sup : super_composite_t;
    
```

В этом случае в адресном пространстве UA сервера будут присутствовать TypeDefinition узлы, изображенные на рисунке 1.

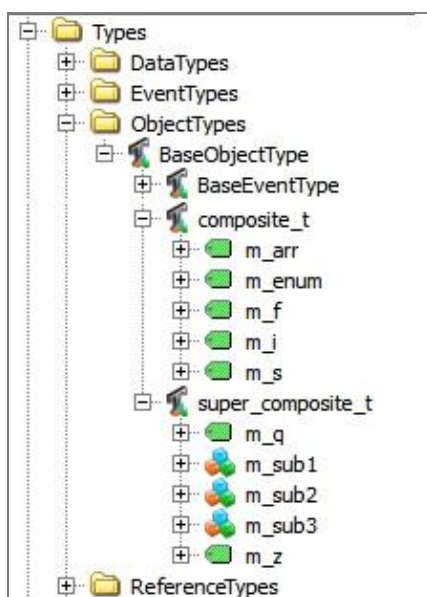


Рисунок 1 - Представление типов данных в адресном пространстве

Все данные IEC-приложений будут располагаться под директорией Root.Objects.IEC_DATA (Рисунок 2).

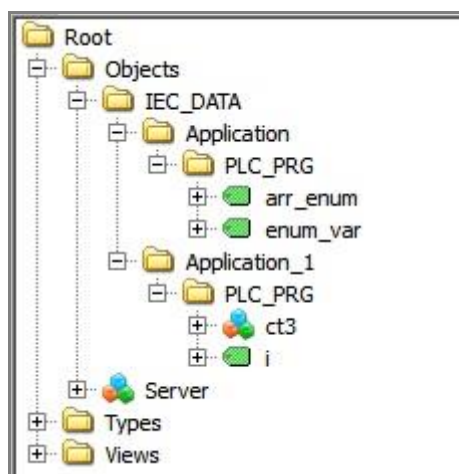


Рисунок 2 - Данные приложений

Конвертация текста в переменных типа STRING

Возможность конвертации русского/английского текста из одной кодировки отображения в другую (ANSI (например, CP1251)⇒UNICODE) в переменных типа STRING осуществляется за счет опции `code_page`.

Настройка задается в файле *ServerConfig.xml* (путь к конфигурационному файлу */etc/OpcUA/ServerConfig.xml*, смотри раздел «Файлы и каталоги, используемые при работе компонента»). Если опция `<code_page>` отсутствует, то конвертации не происходит. Это сделано для того, чтобы не затрагивалась производительность в случаях, когда не используются локальные символы.

Пример задания опции `code_page`:

```
<OpcServerConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <UaServerConfig>
    <code_page>CP1251</code_page>
```

В этом случае PsUaServer будет корректно конвертировать русские символы в UNICODE.

Трансляция данных

Элементарные типы данных

Элементарные типы данных из IEC-приложений транслируются в типы данных протокола UA согласно таблице 1.

Таблица 1 - Отображение элементарных типов данных

| IEC | UA | Node ID | Примечание |
|------|---------|---------|------------|
| BOOL | Boolean | 0:1 | |
| BIT | Byte | 0:3 | |

| IEC | UA | Node ID | Примечание |
|-------------|----------|---------|---|
| BYTE | Byte | 0:3 | |
| WORD | UInt16 | 0:5 | |
| DWORD | UInt32 | 0:7 | |
| LWORD | UInt64 | 0:9 | |
| SINT | SByte | 0:2 | |
| INT | Int16 | 0:4 | |
| DINT | Int32 | 0:6 | |
| LINT | Int64 | 0:8 | |
| USINT | Byte | 0:3 | |
| UINT | UInt16 | 0:5 | |
| UDINT | UInt32 | 0:7 | |
| ULINT | UInt64 | 0:9 | |
| REAL | Float | 0:10 | |
| LREAL | Double | 0:11 | |
| STRING | String | 0:12 | Начиная с версии 1.6.5.0 со стороны UA Server не накладывается дополнительного ограничения на длину строки, ранее декларируемая максимальная длина была 255 |
| WSTRING | String | 0:12 | Начиная с версии 1.6.5.0 со стороны UA Server не накладывается дополнительного ограничения на длину строки, ранее декларируемая максимальная длина была 255 |
| TIME | UInt32 | 0:7 | Количество миллисекунд, прошедших с 00:00 |
| DATE | DateTime | 0:13 | |
| DATEANDTIME | DateTime | 0:13 | |
| TIMEOFDAY | DateTime | 0:13 | Устанавливается только время, дата = текущей |

Специальные структуры для простых переменных

При объявлении переменной простого типа в IEC-приложении:

`i : INT;`

и размещении ее в Symbol Configuration, эта переменная отображается на UA-variable типа Int16.

Метка времени для простых переменных присваивается сервером в момент поступления запроса, кратного интервалу опроса (sampling).

Качество переменной **i** всегда **Good**. Для такой переменной нет способа программно присвоить метку или качество.

Для того, чтобы была возможность задавать метку времени и качество переменным, компонент PsUaServer поддерживает специальные типы данных, определенные в библиотеке PsUaLib.library.

Со стороны ИЕС-приложения данные структуры выглядят как обычные структуры, имеющие компоненты:

```

{
    m_value // тип m_value для каждой определенной структуры ua_var_* свой. см.
таблицу 2.
    m_quality : ua_quality := ua_quality.OpcUA_Good;
    m_timestamp : SysTimeRtc.SysTimeCore.SYSTIME;
}

```

Встретив экземпляр такой структуры в программе, PsUaServer работает с ним определенным образом. В адресном пространстве UA-server экземпляры этих структур выглядят как обычные UA-variable. При этом пользователь имеет возможность программно устанавливать качество и временную метку.

Перечень специальных структур, определенных в PsUaLib с указанием типа данных, представлен в таблице 2.

Таблица 2 - Перечень специальных структур для простых переменных

| Переменная | Тип |
|--------------------|---------------|
| ua_var_bool | BOOL |
| ua_var_byte | BYTE |
| ua_var_date | DATE |
| ua_var_dateandtime | DATE_AND_TIME |
| ua_var_dint | DINT |
| ua_var_dword | DWORD |
| ua_var_int | INT |
| ua_var_lint | LINT |
| ua_var_lreal | LREAL |
| ua_var_ltime | LTIME |
| ua_var_lword | LWORD |
| ua_var_real | REAL |
| ua_var_sint | SINT |
| ua_var_str | STR(80) |
| ua_var_str255 | STR(255) |
| ua_var_time | TIME |

| Переменная | Тип |
|------------------|-------------|
| ua_var_timeofday | TIME_OF_DAY |
| ua_var_udint | UDINT |
| ua_var_uint | UINT |
| ua_var_ulint | ULINT |
| ua_var_usint | USINT |
| ua_var_word | WORD |
| ua_var_wstr | WSTR(80) |
| ua_var_wstr255 | WSTR(255) |
| ua_var_bytestr | BYTESTRING |

Для массивов переменных используются функциональные блоки ua_arr_*** (где *** - bool, byte, word, dword, lword, sint, dint, lint, usint, uint, udint, ulint, real, lreal, int, str, wstr, str255, wstr255, dateandtime, date, ltime, timeofday, time).

Применение специальных структур

Использование специальных структур вместо функциональных блоков значительно уменьшает время загрузки приложения, но накладывает ограничения:

- невозможность автоматической инициализации комплексных переменных в конструкторе, так как структуры не имеют конструкторов;
- невозможность использования property, так как структуры не имеют свойств.

Исходя из этого, для корректной работы комплексных переменных, необходимо выполнить следующие действия:

- 1) В первом цикле программы однократно выполнить действия по инициализации начальных значений комплексных переменных.

Определение переменных, например:

```
my_ua_arr_bool : PsUaLib.ua_arr_bool(size := 10);
my_ua_var_bool : PsUaLib.ua_var_bool;
my_ua_arr_real : PsUaLib.ua_arr_real(size := 20);
my_ua_var_real : PsUaLib.ua_var_real;
```

При объявлении экземпляра ua_arr_*** необходимо указывать size (число элементов массива)

Инициализация переменных в первом цикле:

```
IF ( first_cycle )
THEN
    first_cycle := FALSE;
    my_ua_var_bool.m_value := FALSE;
    my_ua_var_real.m_value := 11.0;
END_IF
```

2) Присвоить значения для комплексных переменных:

```
//пример присвоения значения:  
my_int.m_value := 123;
```

```
//пример присвоения качества:  
my_int.m_quality := PsUaLib.ua_quality.OpcUa_Good;
```

```
// пример "прямого", более быстрого присваивания метки времени:  
SysTimeRtc.SysTimeRtcHighResGet(my_int.m_timestamp);
```

Метка времени будет передана только в случае изменения значения переменной и/или качества.

```
// пример присваивания значений элементам массива:  
FOR ix := 0 TO 9 DO  
    my_ua_arr_bool.m_array[ix] := NOT my_ua_arr_bool.m_array[ix];  
END_FOR  
  
    my_ua_var.m_quality := PsUaLib.ua_quality.OpcUa_Bad;  
    my_ua_arr_bool.m_quality := PsUaLib.ua_quality.OpcUa_Good;
```

Строковые типы определены для строк длиной в 80 символов:

```
ua_var_str : переменная типа STR(80),  
ua_var_wstr : переменная типа WSTR(80)
```

и длиной в 255 символов:

```
ua_var_str255 : переменная типа STR(255),  
ua_var_wstr255 : переменная типа WSTR(255).
```

Поддержка типа ByteString

UA сервер позволяет использовать тип ByteString. ByteString отображается на функциональный блок ua_var_bytestr из библиотеки PsUaLib.library.

Переменная, объявленная как

```
byte_str : PsUaLib.ua_var_bytestr(size := 10);
```

будет отображаться на UA-переменную с типом ByteString.

При объявлении экземпляра ua_var_bytestring необходимо указывать размер в байтах.

```
( size := 10 );
```

Максимальный размер строки в байтах 65535.

Можно устанавливать байты с помощью метода set_byte().

Например:

```
byte_str.set_byte(0, 33);
```

Ниже представлен интерфейс метода:

```
METHOD set_byte : BOOL  
VAR_INPUT
```

```
ix : UINT; // zero-based индекс массива
val : BYTE; // значение
END_VAR
```

Метод возвращает значение типа BOOL. TRUE если установка значения была успешна, FALSE в обратном случае (например, задан недопустимый индекс).

Можно читать байты с помощью метода `get_byte()`.

Например:

```
bt : BYTE;

byte_str.get_byte(0, bt);
```

Ниже представлен интерфейс этого метода:

```
METHOD get_byte : BOOL
VAR_INPUT
  ix : UINT; // zero-based индекс массива
  val : REFERENCE TO BYTE; // прочитанное значение присваивается в val
END_VAR
```

Метод возвращает значение типа BOOL. TRUE если чтение было выполнено успешно, FALSE в обратном случае (например, задан недопустимый индекс).

Трансляция перечислимого типа

Если в IEC приложении определен перечислимый тип (ENUMERATED):

```
TYPE COLOR_ENUM_TYPE :
(
  RED := 10,
  GREEN := 20,
  YELLOW := 30,
  BLUE := 40,
  BROWN := 50
);
END_TYPE
```

то в UA сервере будет сгенерирован тип данных, соответствующий данному перечислению. Располагаться он будет в адресном пространстве в виде узла:

`Types.DataTypes.BaseDataType.Enumeration.COLOR_ENUM_TYPE`.

Отображение массивов структур в адресном пространстве UA сервера

Если в приложении, работающем на контроллере, определены структуры `composite_t` `super_composite_t` из предыдущих примеров, а также имеются массивы, состоящие из этих структур:

```
arr_comp : ARRAY[1..5] OF composite_t;
arr_sup_comp : ARRAY[1..10] OF super_composite_t;
```

то в UA сервере появятся определения типов, сгенерированные для таких данных. Пример типов данных, сгенерированных для массивов структур, а также пример данных массивов структур приведены на рисунке 3.

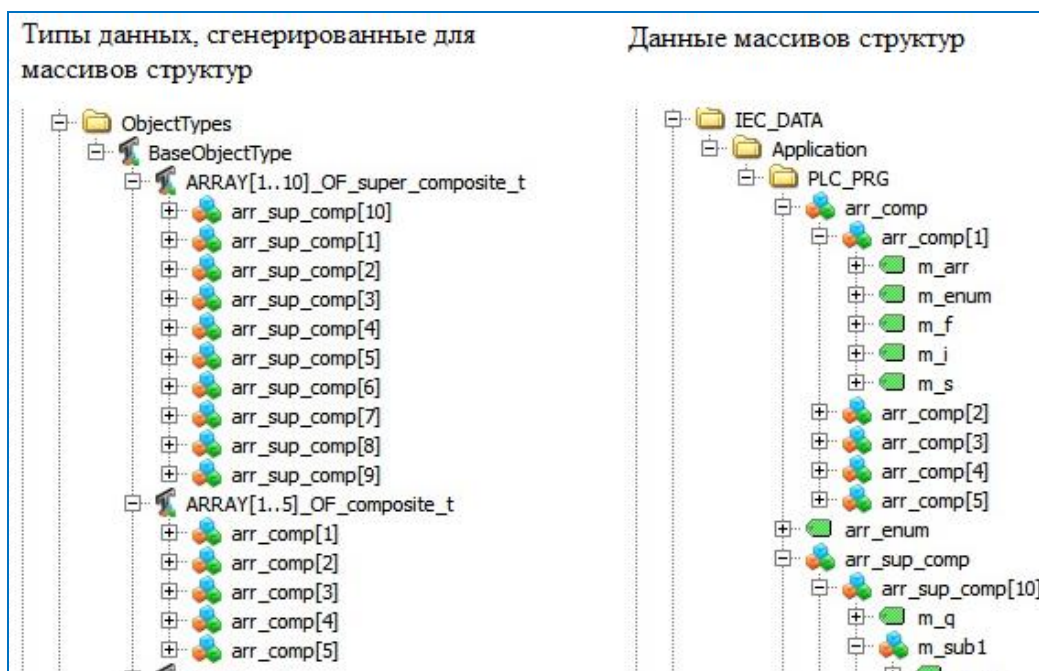


Рисунок 3 - Типы данных и данные

Массивы элементарных данных представляются согласно стандартной модели данных UA. В переменной (Variable), представляющей массив, атрибут ValueRank равен 1, а атрибут Value содержит массив заданной размерности, состоящий из элементарных типов.

Отображение статических переменных в адресном пространстве UA сервера

Статические переменные определяются между ключевыми словами VAR_STAT и END_VAR. В отличие от переменных-членов функциональных блоков, которые содержатся в каждом отдельном экземпляре, статические переменные содержатся в определении функционального блока в единственном числе.

В адресном пространстве UA Server статические переменные фигурируют как в определении функционального блока, так и в каждом отдельном экземпляре функционального блока. Например:

```
FUNCTION_BLOCK FB_t
VAR
    simple_var : ULINT;
END_VAR
VAR_STAT
    z_static : ULINT;
END_VAR
```

В приложении имеется 2 экземпляра FB_t

```
PROGRAM PLC_PRG
VAR
    v1 : FB_t;
    v2 : FB_t;
END_VAR
```

В этом случае OPC UA клиент в адресном пространстве будет представлен, как показано на рисунке 4.

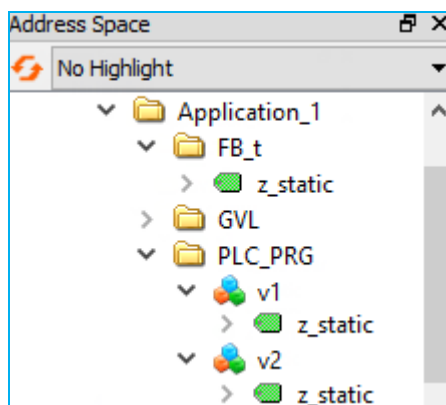


Рисунок 4 - Статические переменные

При этом все три имеющиеся `z_static` будут ссылаться на одну статическую переменную.

И при изменении `v1.z_static` будут изменяться так же `v2.z_static`, `FB_t.z_static`.

КОНФИГУРИРОВАНИЕ СЕРВЕРА OPC UA

Подключение к ПЛК

Для начала работы с конфигуратором необходимо подключиться к контроллеру через сканер сети (см. «Программное обеспечение Astra.IDE. Руководство пользователя», раздел «Подключение контроллера к сети»).




ВНИМАНИЕ!

Порты коммуникационного модуля CP xx 021 не поддерживают работу по протоколу OPC UA

Включение OPC UA сервера

По умолчанию сервер OPC UA отключен, поэтому необходимо запустить сервер, выполнив следующие действия:

- в Astra.IDE на главной вкладке параметров устройства перейдите на вкладку **Сервис ПЛК** ⇒ **Системные параметры**. Далее нажмите кнопку  (**Обновить**). На экран будет выведена информация о текущем состоянии доступных параметров (Рисунок 5);

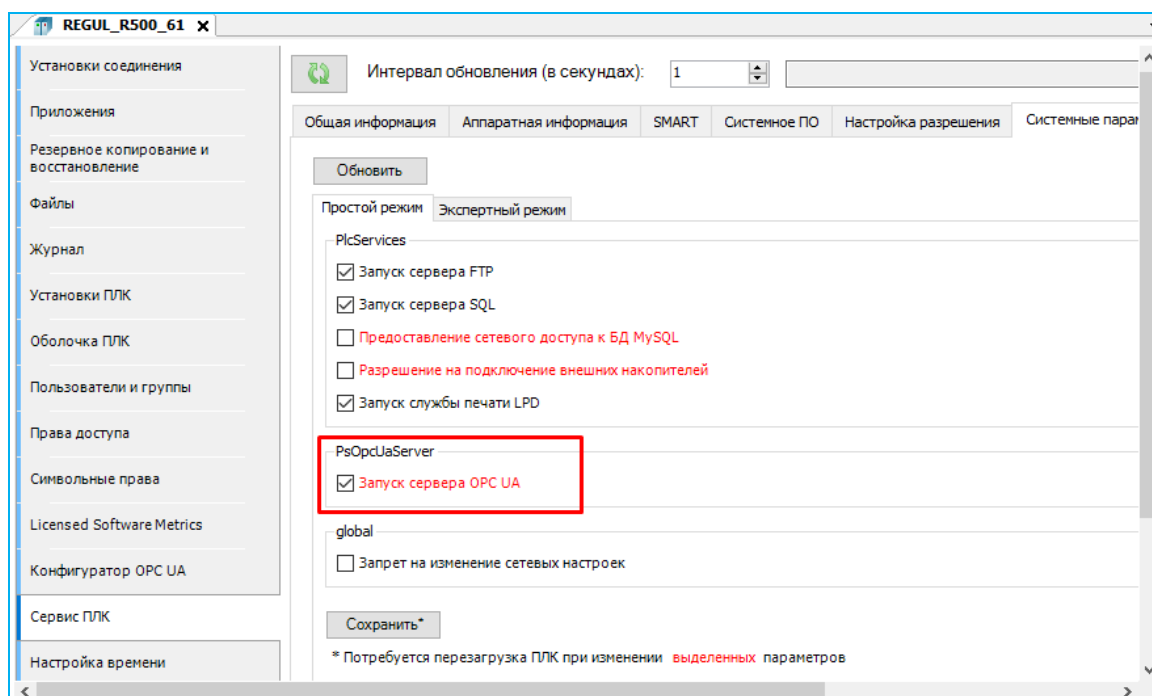


Рисунок 5 – Вкладка с системными параметрами в простом режиме. Включение сервера OPC UA



ВНИМАНИЕ!

В зависимости от версии СПО по умолчанию будет запускаться следующая версия OPC UA сервера:

- начиная с версии СПО 1.7.0.0 – **OpcUaServer_OS**;
- до версии СПО 1.7.0.0 – **OpcUaServer**

– для активации:

- в простом режиме: установите флажок в поле **Запуск сервера OPC UA** и нажмите кнопку **Сохранить**;
- в экспертном режиме: выберите название каталога конфигурационного файла - **etc/runtime.cfg** и в секции **[PsOpcUaServer]** добавьте соответствующую строку:

До версии СПО 1.7.0.0

▶ для включения **OpcUaServer**

```
[PsOpcUaServer]  
Enable=1
```

▶ для включения **OpcUaServer_OS**

```
[PsOpcUaServer]  
EnableV2=1
```

Начиная с версии СПО 1.7.0.0

▶ для включения **OpcUaServer**

```
[PsOpcUaServer]  
EnableLegacy=1
```

▶ для включения **OpcUaServer_OS**

```
[PsOpcUaServer]  
Enable=1
```



ВНИМАНИЕ!

Разрешается активировать только одну из версий!

– перезагрузите контроллер (путем выключения/включения питания или командой *reboot* на вкладке **Оболочка ПЛК**).

Описание конфигуратора

Конфигуратор OPC UA, встроенный в среду разработки Astra.IDE, реализует стандартный интерфейс доступа к данным.

Конфигуратор предоставляет следующие возможности (Рисунок 6):

- выгрузка сертификата сервера с контроллера (путь хранилища сертификатов `etc /.../ own /certs`) (**Download certificate from controller...**);
- добавление / удаление сертификатов сервера / клиентские:
 - управление сертификатами эмитентов (издателей) (**ISSUERS CERTS...**);
 - управление списком отозванных сертификатов (**CERT REVOCATION LIST...**);
 - управление доверенными сертификатами (**TRUSTED CERTS...**);
 - управление списком доверенных отозванных сертификатов (**TRUSTED REVOCATION LIST...**);

- настройка:
 - параметров ведения журналов работы сервера (**Трассировка**);
 - параметров точек подключения (**Точка подключения**);
 - параметров уровня доступа пользователей (**Пользователи**);
 - включение/выключение отправки определенных событий (**События**).

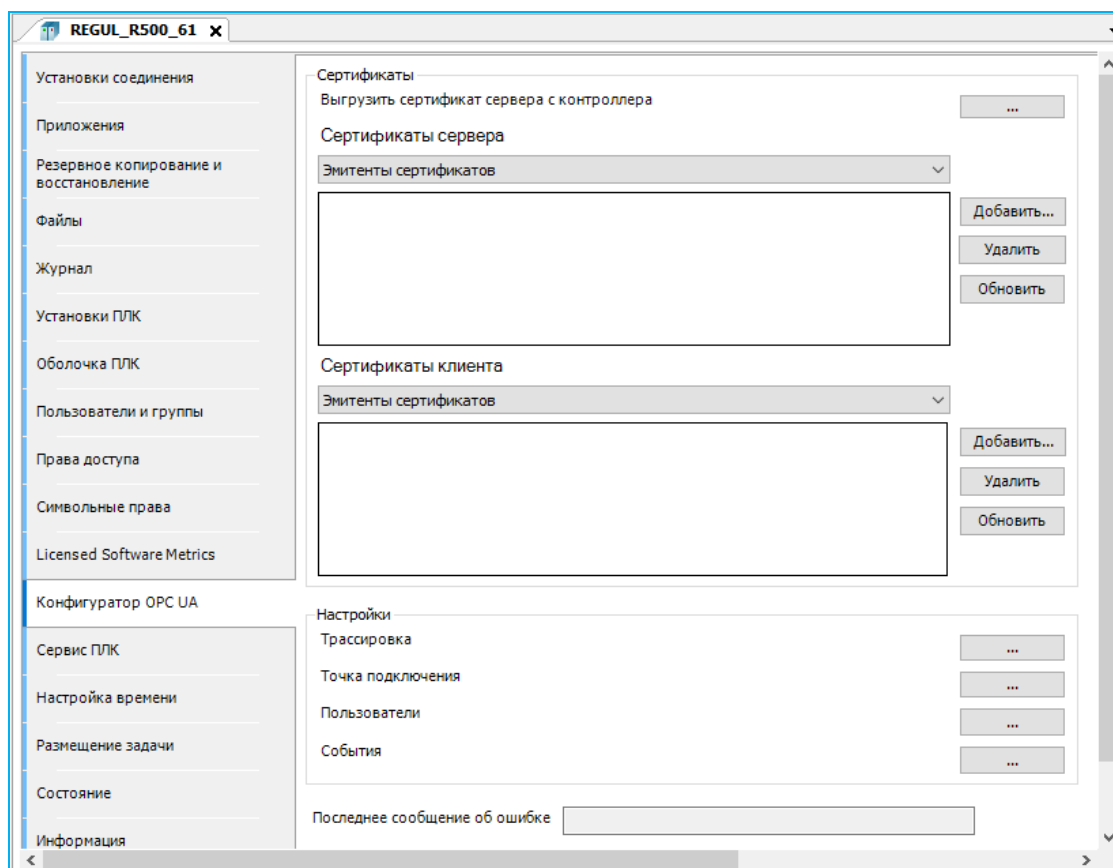



Рисунок 6 - Вкладка конфигуратора OPC UA

Способы аутентификации:

- **Сертификат безопасности** – для авторизации используется путь к файлу X509v3-сертификата (смотри раздел «Настройка соединений с OPC UA клиентами»);
- **Имя пользователя и пароль** – для авторизации на OPC-сервере используется имя пользователя и пароль (смотри абзац «Строка **Пользователи**»);
- **Анонимный (Anonymous)** – информация о пользователе недоступна (смотри раздел «Ограничение анонимного доступа к серверу OPC UA»).

Поле «Сертификаты»

Для установления соединения между клиентом и сервером OPC UA необходимо, чтобы сертификат сервера был в списке доверенных сертификатов клиента (TRUSTED CERTS), а сертификат клиента был в списке доверенных сертификатов сервера (TRUSTED CERTS). Для этого выполните следующие действия:

-  `regul.der` - выгрузите сертификат сервера (путь хранения - `etc /.../ server / own / certs`) из ПЛК с помощью кнопки (указав путь сохранения) для добавления его в группу доверенных сертификатов клиента (TRUSTED CERTS);
- загрузите в ПЛК сертификат клиента (TRUSTED CERTS) и / или эмитенты сертификаты ЦС (ISSUERS CERTS).



ИНФОРМАЦИЯ

Место хранения сертификатов сервера на ПЛК:

- `etc /.../ own / certs` – публичный сертификат *.der,
- `etc /.../ own / private` – закрытый сертификат *.pem

Поле «Настройки»

Regul OPC UA сервер ведет два журнала событий:

- первый журнал: сообщения трассировки стека UA и UA сервера. В этот журнал выводятся сообщения о работе стека UA и UA сервера.
- второй журнал: сообщения о состоянии и работе сервера UA, которые отображаются в журнале Astra.IDE (`StdLogger.log`). В этот журнал выводятся сообщения о работе приложения ПЛК (запуск/останов приложения, отладочные сообщения об инициализации переменных etc)



ИНФОРМАЦИЯ

Новый OPC UA сервер (OpcUaServer_OS) дополнительно журналирует информацию в файл `opcua_dhub_driver.log` (путь к файлу `logs/logger/user/`)

Строка Трассировка

С помощью кнопки откройте окно **Настройки трассировки** (Рисунок 7).

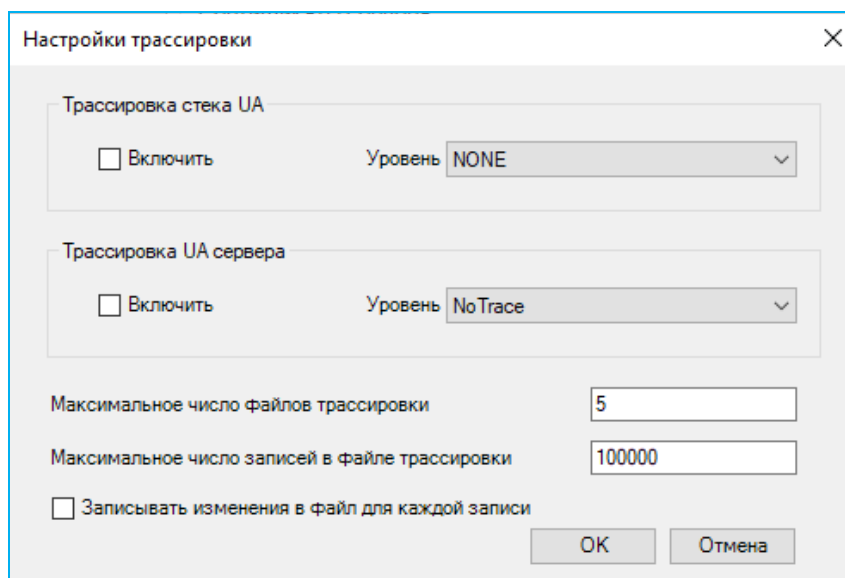


Рисунок 7 - Окно настройки трассировки

Для настройки трассировки стека UA установите в соответствующем поле флажок **Включить** (по умолчанию трассировка выключена), что позволит изменить уровень детализации журналирования (см. таблицу 3). В поле **Уровень** (для фильтрации событий) выберите значение из выпадающего списка, показывающее, какие сообщения будут выводиться в журнал.

Таблица 3 – Уровни журналирования стека UA

| Уровень | Описание |
|---------|--|
| NONE | Сообщения не выводятся |
| ERROR | Выводятся сообщения об ошибке, формируемые при обработке исключительных ситуаций |
| WARNING | Предупреждения о возникновении нежелательной ситуации, требуется обратить внимание |
| SYSTEM | Сообщения от системных служб |
| INFO | Общие информационные сообщения о том, что происходит. Нормальный ход работы |
| DEBUG | Сообщения, используемые во время отладки |
| CONTENT | Более подробные сообщения (полный текст), используемые во время отладки |
| ALL | Все сообщения |

Для настройки трассировки сервера UA установите в соответствующем поле флажок **Включить** (по умолчанию трассировка выключена). В поле **Уровень**, из выпадающего списка, выберите необходимую полноту информации для журналирования (см. таблицу 4).

Таблица 4 – Уровни журналирования сервера UA

| Уровень | Описание |
|---------------|---|
| No Trace | Журналирование отсутствует |
| Errors | Выводятся сообщения об ошибке |
| Warning | Предупреждения о возникновении нежелательной ситуации |
| Info] | Информационные сообщения о работе |
| IntarfaseCall | Вызовы к интерфейсам модуля |
| CtorDtor | Создание и уничтожение объектов |
| ProgramFlow | Внутренний поток программы |
| Data | Данные |

Для записи сообщений трассировки в журнал необходимо задать расположение и название файла журнала в приведенном ниже параметре файла конфигурации (путь к файлу *etc/OpcUA/ServerConfig.xml*):

```
<UaServerConfig>
<Trace>
<UaAppTraceFile>/mnt/user/archive/logs/regul_ua_server.log</UaAppTraceFile>
```

Созданный файл будет расположен в *logs/regul_ua_server.log*.

Запись в текущий файл трассировки стека и сервера заканчивается, когда число сообщений превысит значение параметра **UaAppTraceMaxEntries** (**Максимальное число записей в файле трассировки**). После этого создается новый файл трассировки с именем, отличающимся на цифровое окончание (индекс). Создание новых файлов трассировки заканчивается, когда превысит значение параметра **UaAppTraceMaxBackup** (**Максимальное число файлов трассировки**).

Например, для заданных параметров:

```
<UaAppTraceMaxEntries>100000</UaAppTraceMaxEntries>
<UaAppTraceMaxBackup>5</UaAppTraceMaxBackup>
```

Запись сообщений трассировки будут вестись последовательно в 5 файлов (значение параметра **UaAppTraceMaxBackup**):

```
regul_ua_server.log
regul_ua_server_1.log
regul_ua_server_2.log
regul_ua_server_3.log
regul_ua_server_4.log
```

После того как число сообщений в файле *regul_ua_server_4.log* превысит 100000 (значение параметра **UaAppTraceMaxEntries**), запись сообщений будет снова производиться в файл *regul_ua_server.log* (по кольцу).

Строка Точка подключения

С помощью кнопки  откройте окно **Настройки точек подключения** (Рисунок 8).

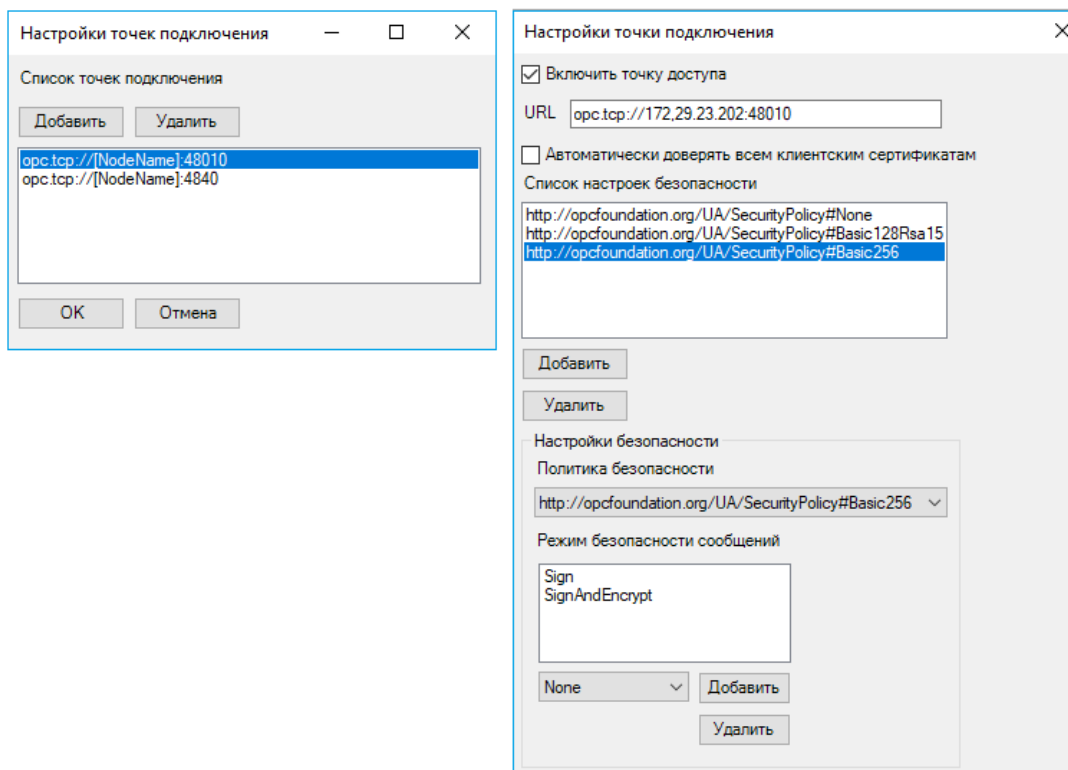


Рисунок 8 – Настройка параметров точек подключения

Точек соединения может быть несколько и для добавления/удаления используйте, соответственно, кнопки **Добавить** и **Удалить**.

Перейти, к редактированию параметров точки соединения, можно двойным щелчком левой кнопкой мыши по нужной строке. Появится дополнительное окно. Поставьте флажок в поле **Включить точку доступа**. В поле **URL** вместо [nodeName] введите IP-адрес контроллера, по которому будет устанавливаться соединение (порт для открытия входящих TCP – соединений: 48010/4840). Установка флажка в строке **Автоматически доверять всем клиентским сертификатам** позволяет всем сертификатам с данного контроллера автоматически наследовать доверие.



ВНИМАНИЕ!

Самым простой способом подключения является подключение к серверу анонимно и без политики безопасности (если сервер это допускает). Это наиболее простой и небезопасный вариант подключения, который не рекомендуется использовать

В области **Список настроек безопасности** выберите режим политики безопасности *http://opcfounddtion.org/UA/SecurityPolicy#...*:

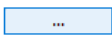
- без политики безопасности – *None*;
- с политикой безопасности:

- *Basic128Rsa15* – базовый 128-разрядный алгоритм шифрования сообщений (устарел и больше не считается безопасным);
- *Basic256* – базовый 256-разрядный алгоритм шифрования сообщений.

После выбора алгоритма шифрования, в области **Режим безопасности сообщений**, выберите один из двух режимов безопасности сообщений:

- *Sign* – подписать все сообщения, но не шифровать их;
- *Sign and encrypt* – подписать и зашифровать все сообщения.

Строка Пользователи

К профилю пользователя можно привязывать объекты, к которым он будет иметь доступ для управления и мониторинга параметров. Напротив строки **Пользователи** нажмите кнопку , чтобы ограничить доступ паролем для удаленных клиентов – в этом случае после запуска клиента выведется окно для ввода пароля (Рисунок 9).

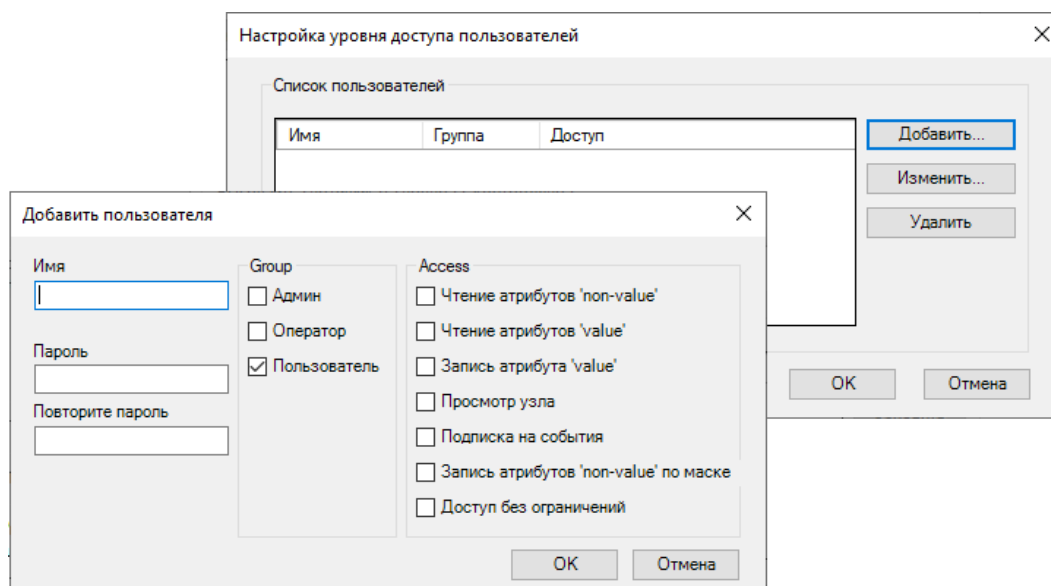


Рисунок 9 – Настройка уровня доступа пользователей

Строка События

Для отключения отправки определенных событий снимите с соответствующего поля флажок (по умолчанию включено). Возможно формирование следующих событий (Рисунок 10):

- **«Before deletion»** - событие, генерирующееся перед удалением приложения;
- **«Prepare download»** - событие при загрузке приложения;
- **«Started»** - событие при запуске (перезапуске) приложения (для впервые стартующих приложений отправляется всегда, вне зависимости от установки/снятия галочки);
- **«Stopped»** - событие при остановке работающего приложения.

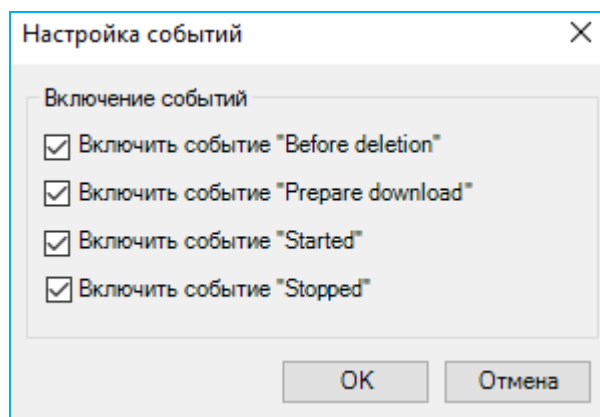


Рисунок 10 - Настройка событий

Для вступления в силу изменений потребуется перезагрузить контроллер путем выключения/включения питания либо командой *reboot* на вкладке **Оболочка ПЛК**.

Настройка соединений с OPC UA клиентами

При аутентификации, установлении защищенного соединения и обмене сообщениями, UA-приложения используют сертификаты X509 Version 3, закодированные в формате DER (*.der). Сервер автоматически генерирует самоподписанные сертификаты контроллера. Для успешного подключения UA клиенту необходимо импортировать в группу своих доверенных сертификатов (TRUSTED) самоподписанные сертификаты.

Этого достаточно для верификации сертификата, предоставляемого сервером клиенту на этапе установления соединения.

Ограничение анонимного доступа к серверу OPC UA

Включение/отключение анонимного доступа к серверу OPC UA производится через конфигурационный файл *ServerConfig.xml*. Для работы с конфигурационным файлом *ServerConfig.xml* выполните следующие действия:

- перейдите на вкладку **Файлы** в Astra.IDE и пройдите по указанному пути */etc/OpcUA/ServerConfig.xml* (Рисунок 11);

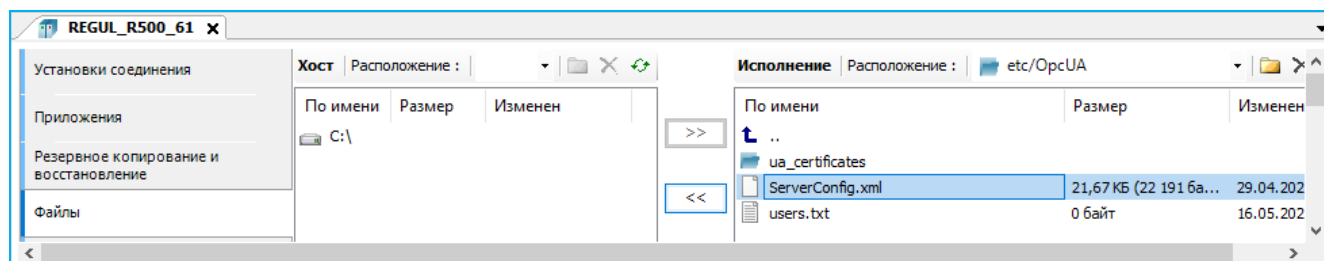



Рисунок 11 - Конфигурационный файл UA сервера, расположенный в папке etc на контроллере

- кнопкой  скопируйте файл *ServerConfig.xml* с контроллера на ПК (из **Исполнение** в **Хост**);
- откройте на ПК файл *ServerConfig.xml*;

- сама конфигурация поддерживаемых токенов идентификации пользователя хранится в элементе `<UserIdentityTokens>` с возможными значениями – *true* (по умолчанию) или *false* (Рисунок 12). Тип `tokenType` со значением `Anonymous` указывает, что Сервер не требует какой-либо идентификации пользователя;

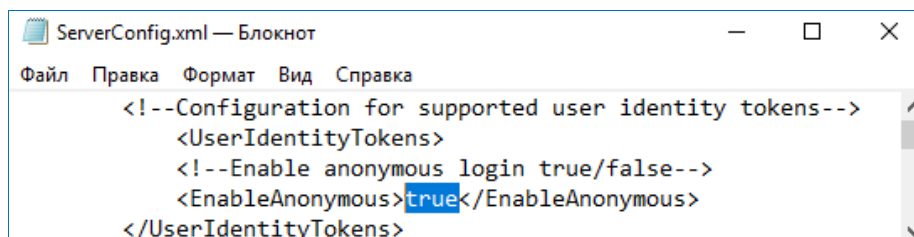




Рисунок 12 – Изменение конфигурации идентификации пользователя

- измените значение в строке с *true* → *false* для закрытия анонимного доступа;
- сохраните изменения в файле `ServerConfig.xml`;
- в Astra.IDE на вкладке **Файлы** кнопкой  скопируйте измененный файл с ПК на контроллер (из **Хост** в **Исполнение**);
- перезагрузите контроллер (путем выключения/включения питания или командой `reboot` на вкладке **Оболочка ПЛК**).

Теперь, при попытке подключения анонимного пользователя (`Anonymous`), будет всплывать ошибка «`BadIdentityTokenRejected`».

Файлы и каталоги, используемые при работе компонента

Все настройки компонента OPC UA, список пользователей, файлы сертификатов находятся в директории: `etc/OpcUA`.

В Astra.IDE на главной вкладке параметров устройства перейдите на вкладку **Файлы**. В области **Исполнение** нажмите кнопку  (**Обновить**). В окне отобразится дерево файлов, имеющихся на контроллере. Найдите папку `etc` → **OpcUA** (Рисунок 13).

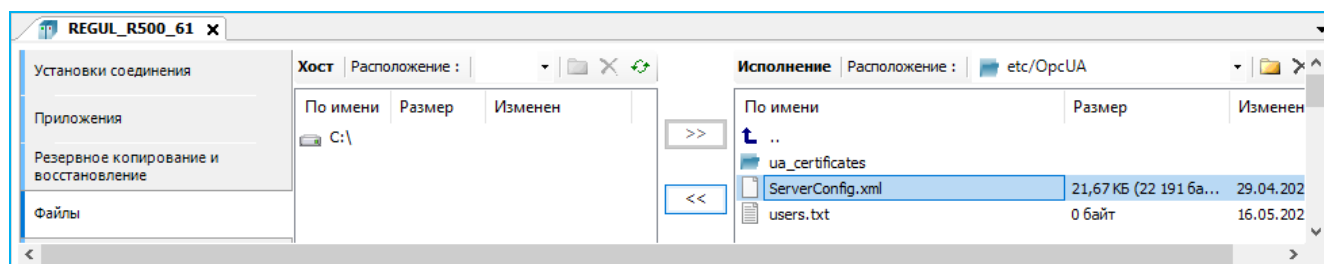


Рисунок 13 - Расположение каталогов компонента OPC UA на контроллере

Путь к конфигурационному файлу UA сервера `/etc/OpcUA/ServerConfig.xml`.

Путь к хранилищу сертификатов и спискам отзывает `/etc/OpcUA/ua_certificates/...`

Путь к файлу описания пользователей `/etc/OpcUA/users.txt`.

Структура каталогов **OPC UA** на контроллере показана ниже (Рисунок 14).

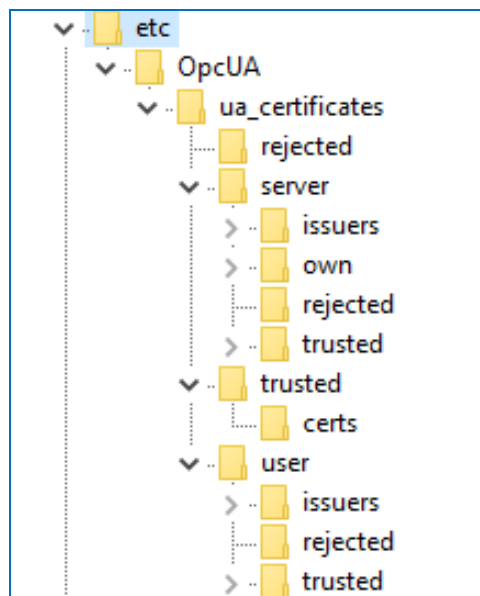


Рисунок 14 - Структура каталогов компонента OPC UA на контроллере

Добавление переменных в адресное пространство сервера OPC UA

Для организации передачи данных по протоколу OPC UA, необходимо в программе Astra.IDE добавить **Symbol Configuration**. Для этого в контекстном меню приложения (**Application**) выберите **Добавить объект** (Add object) → **Символьная конфигурация...** (Рисунок 15).

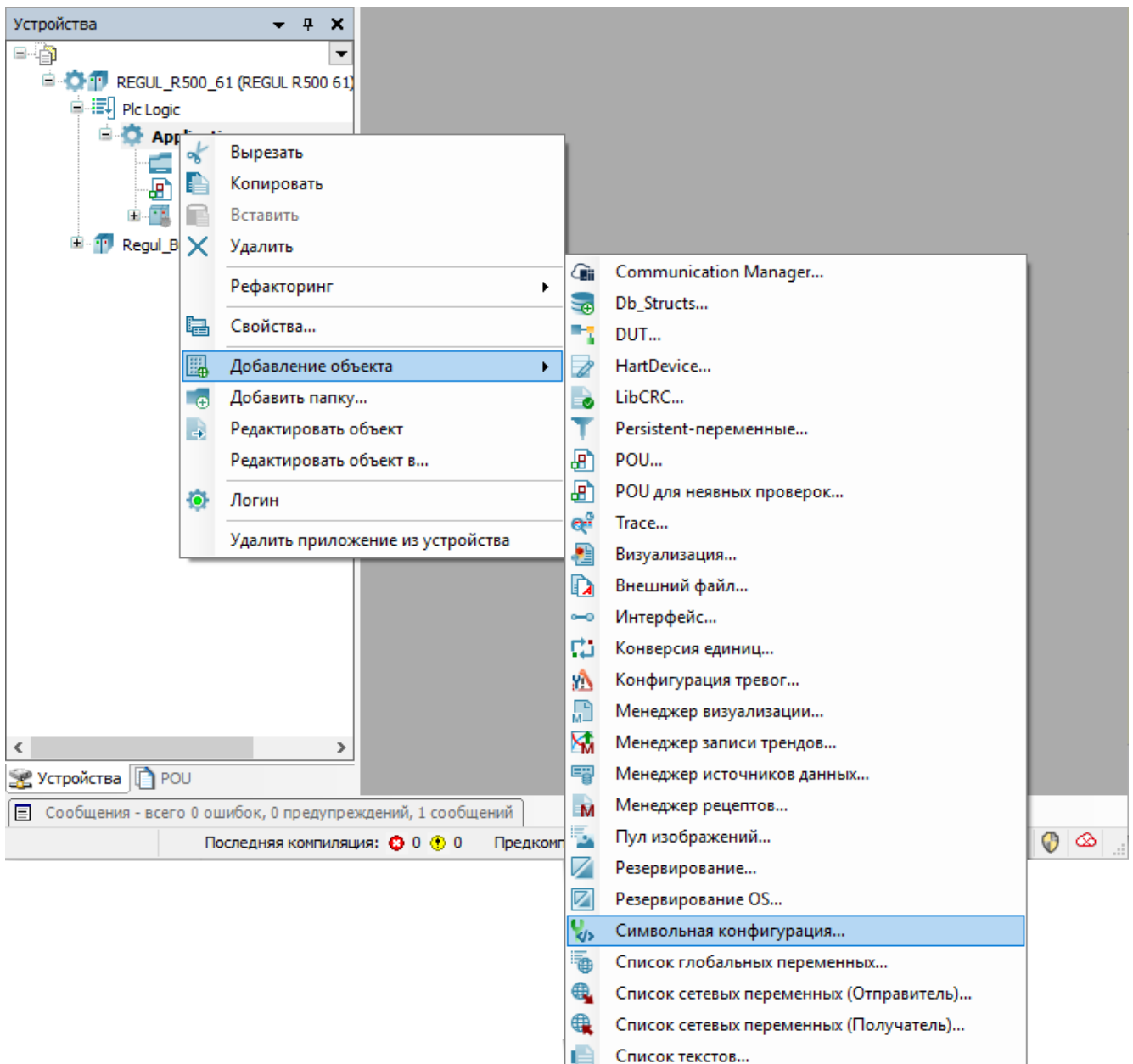


Рисунок 15 - Контекстное меню

Откроется окно **Добавить Символьная конфигурация**, где установите флажок в поле **Поддержка функций OPC UA** и нажмите кнопку **Добавить** (Рисунок 16).

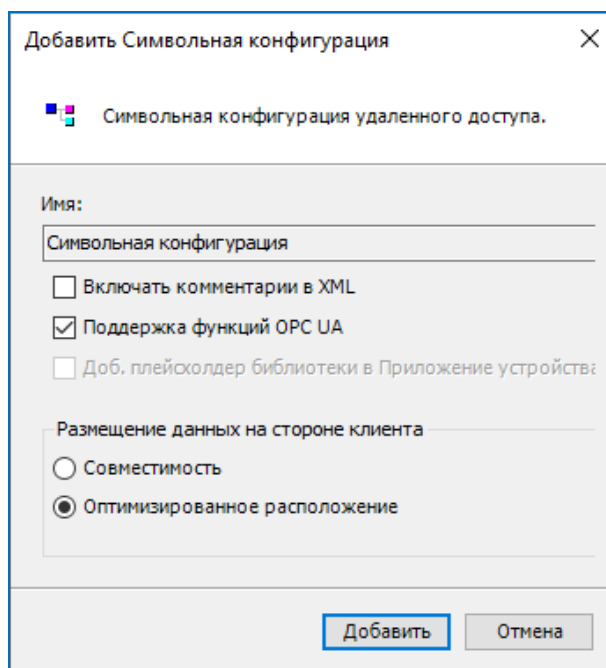


Рисунок 16 - Добавление Символьной конфигурации

Выберите в дереве устройств появившийся объект **Символьная конфигурация** и двойным щелчком мыши по названию откройте вкладку. Если при добавлении пропустили настройку **Поддержка функций OPC UA**, выберите закладку **Установки** и установите флажок в соответствующем поле (Рисунок 17).

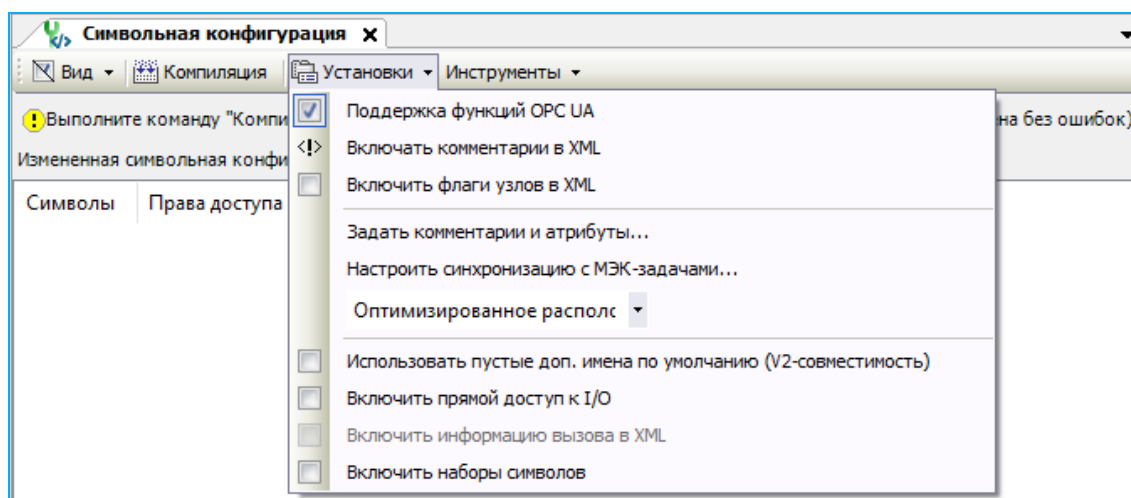



Рисунок 17 - Дополнительное подтверждение необходимости поддержки символьной конфигурации

Изначально на вкладке может высветиться сообщение со знаком , сообщающее о необходимости выполнить компиляцию для выявления ошибок (Рисунок 18).

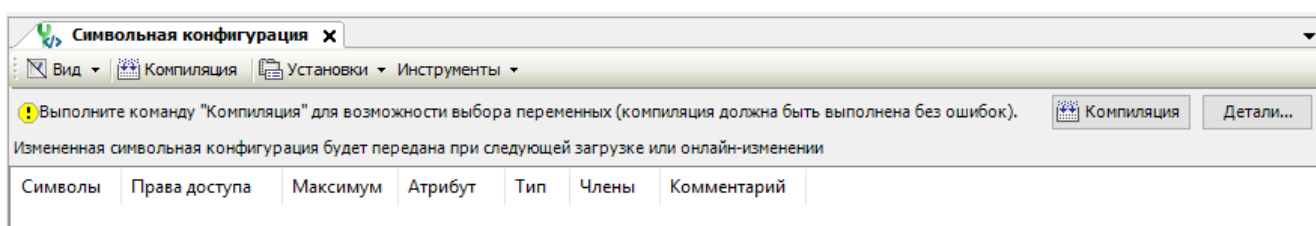


Рисунок 18 – Сообщение о предупреждение

При успешной компиляции отобразится дерево папок. Из дерева папок выберите нужную (PLC_PRG, GVC и т.д.) и в ней раскройте список переменных для добавления, определенных в ИЕС-приложении (Рисунок 19). Установите флажок напротив тех переменных, взаимодействие с которыми будет обеспечиваться протоколом OPC UA.

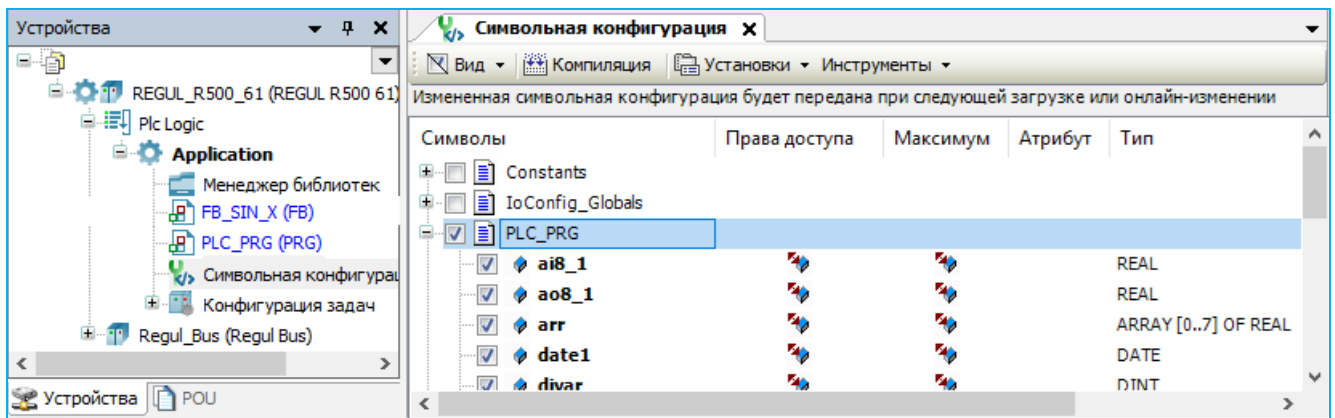


Рисунок 19 - Добавление переменных

СОБЫТИЯ, ГЕНЕРИРУЕМЫЕ OPC UA СЕРВЕРОМ

Генерируемые сервером OPC UA события (Event Notifications, в дальнейшем просто события) информируют UA клиентов об изменении состояния PLC-приложения. Все генерируемые сервером события имеют определенный в спецификации OPC UA тип **GeneralModelChangeEvent** (NodeId = 0:2133).

События генерируются сервером в следующих случаях:

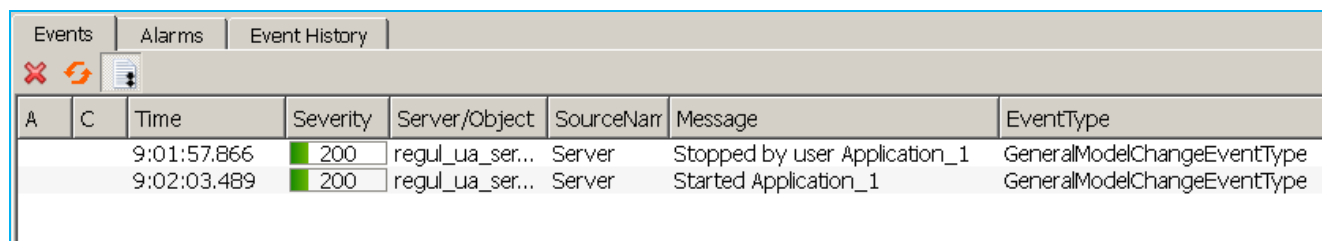
- останов контроллера переключателем (STOP);
- старт контроллера переключателем (RUN);
- останов работающего приложения с помощью меню **Application⇒Stop**;
- запуск остановленного приложения с помощью меню **Application⇒Start**;
- выгрузка приложения с помощью меню **Delete application from device**;
- при выполнении процедуры **Логин с онлайн-изменением** и **Логин с загрузкой**;
- запуск приложения.

Есть возможность включать/выключать отправку определённых событий OPC UA (см. раздел «Описание конфигуратора. Поле «Настройки» пункт «Строка События»).

В последующих приведенных примерах предполагается, что приложение имеет имя **«Application_1»**.

Сообщения событий:

- при останове контроллера переключателем: сообщение **«Switched stop Application_1»**;
- при старте контроллера переключателем: сообщение **«Started Application_1»**;
- при останове работающего приложения с помощью меню **Application⇒Stop**: сообщение **«Stopped by user Application_1»**;
- при запуске остановленного приложения с помощью меню **Application⇒Start**: сообщение **«Started Application_1»**;

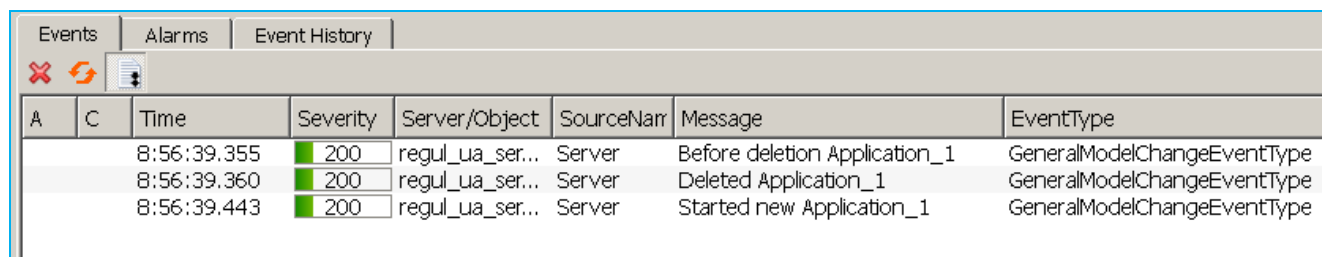


| A | C | Time | Severity | Server/Object | SourceName | Message | EventType |
|---|---|-------------|----------|-----------------|------------|-------------------------------|-------------------------|
| | | 9:01:57.866 | 200 | regul_ua_ser... | Server | Stopped by user Application_1 | GeneralModelChangeEvent |
| | | 9:02:03.489 | 200 | regul_ua_ser... | Server | Started Application_1 | GeneralModelChangeEvent |

Рисунок 20 – Сообщение при запуске/останове приложения с помощью меню в Astra.IDE

- при выгрузке приложения с помощью меню **Delete application from device**: сообщение **«Before deletion Application_1»**, **«Deleted Application_1»**.
- при запуске приложения после загрузки: сообщение **«Started new Application_1»**;

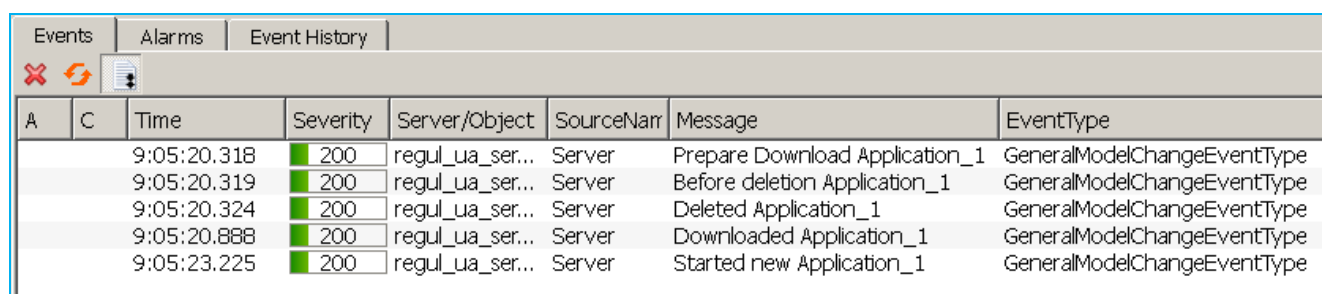
- при выполнении процедуры **Логин с онлайн-изменением**: сообщения «**Before deletion Application_1**» → «**Deleted Application_1**» → «**Started new Application_1**» (Рисунок 21);



| A | C | Time | Severity | Server/Object | SourceName | Message | EventType |
|---|---|-------------|----------|-----------------|------------|-------------------------------|-------------------------|
| | | 8:56:39.355 | 200 | regul_ua_ser... | Server | Before deletion Application_1 | GeneralModelChangeEvent |
| | | 8:56:39.360 | 200 | regul_ua_ser... | Server | Deleted Application_1 | GeneralModelChangeEvent |
| | | 8:56:39.443 | 200 | regul_ua_ser... | Server | Started new Application_1 | GeneralModelChangeEvent |

Рисунок 21 - Сообщения при выполнении процедуры «Логин с онлайн-изменением»

- при выполнении процедуры **Логин с загрузкой**: сообщения «**Prepare Download Application_1**» → «**Before deletion Application_1**» → «**Deleted Application_1**» → «**Downloaded Application_1**» → «**Started new Application_1**» (Рисунок 22);



| A | C | Time | Severity | Server/Object | SourceName | Message | EventType |
|---|---|-------------|----------|-----------------|------------|--------------------------------|-------------------------|
| | | 9:05:20.318 | 200 | regul_ua_ser... | Server | Prepare Download Application_1 | GeneralModelChangeEvent |
| | | 9:05:20.319 | 200 | regul_ua_ser... | Server | Before deletion Application_1 | GeneralModelChangeEvent |
| | | 9:05:20.324 | 200 | regul_ua_ser... | Server | Deleted Application_1 | GeneralModelChangeEvent |
| | | 9:05:20.888 | 200 | regul_ua_ser... | Server | Downloaded Application_1 | GeneralModelChangeEvent |
| | | 9:05:23.225 | 200 | regul_ua_ser... | Server | Started new Application_1 | GeneralModelChangeEvent |

Рисунок 22 - Сообщения при выполнении процедуры «Логин с загрузкой»

- при запуске предварительно остановленного приложения: сообщение «**Started Application_1**».

Следует подчеркнуть отличие между событиями с сообщениями «**Started Application_1**» и «**Started new Application_1**».

Событие с сообщением «**Started Application_1**» генерируется в случае, когда приложение было остановлено любым способом, но не выгружалось из контроллера, и после этого было запущено вновь. В этом случае адресное пространство контроллера не меняется и клиенту не обязательно выполнять обновление подписок на данные.

Событие с сообщением «**Started new Application_1**» генерируется в случае, когда приложение загружено в первый раз, либо оно было выгружено из контроллера. В этом случае адресное пространство контроллера меняется и клиенту необходимо выполнить обновление подписок на данные.

При остановке приложения любым способом качество данных меняется на **BadOutOfService**.

При выгрузке/остановке приложения качество данных меняется на **BadResourceUnavailable**.

Клиент, подключенный к серверу PsOpcUaServer, должен использовать данные события для корректной работы с данными.

Если клиент использует функциональность «**Browse**», то выполнять операции получения (обновления) адресного пространства приложения можно только после получения события **Started new Application_1**» или «**Started Application_1**».

Если приложение было полностью выгружено (сообщение в событии «**Deleted Application_1**») и клиент был подписан на данные, то после повторного запуска приложения, клиенту следует произвести действия по «переподписке» на данные. То есть, удалить из подписки **Item** имеющие качество **BadResourceUnavailable** и добавить в подписку данные с теми же **NodeId**.

В случае, когда приложение не было полностью выгружено из контроллера, а остановлено любым способом, то выполнять операцию "переподписки" не требуется - данные в подписке снова станут актуальными после получения клиентом сообщения «**Started Application1**».

УСТРАНЕНИЕ НЕПОЛАДОК

Для анализа и диагностики работы компонента предусмотрено ведение журнала его работы, подробность сообщений определяется его конфигурацией. Файлы журнала работы компонента сохраняются на контроллере в директории журналов работы компонентов контроллера. Получить эти файлы можно, подключившись к контроллеру FTP-клиентом по адресу <ftp://plclogs:service@plc/>, где plc – адрес контроллера.

Клиент не может установить соединение с сервером

Признаки данного состояния: UA сервер работает, в журнале нет ошибок, но клиент не может установить соединение.

Чаще всего клиент не может установить соединение по причине того, что сертификат клиента не входит в число доверенных. Определить это можно, проверив каталог **ua_certificates/rejected**. Возможно, что этот каталог содержит вновь появившийся файл с сертификатом клиента. Если это так, то самым простым решением данной проблемы будет перенос файла сертификата клиента из каталога **rejected** в каталог **trusted**. Точно так же возможна ситуация, когда сертификат PsOpcUaServer не входит в число сертификатов, которым доверяет используемый UA клиент. В этом случае следует также проверить каталог **rejected** хранилища сертификатов клиента и выполнить действия по внесению сертификата PsOpcUaServer в список доверенных.

Не отображаются пользовательские переменные

Признаки данного состояния: клиент успешно подключился к UA серверу, системные переменные, находящиеся в каталоге **Root.Objects.Server** видны, но каталог **Root.Objects.IEC_DATA** пуст.

Возможная причина заключается в том, что в IEC-приложении в разделе **Symbol Configuration** пользователь не отметил переменные галочками, как доступные для VarAccess. В этом случае надо остановить приложение, перекомпилировать его, отметить переменные как доступные в разделе **Symbol Configuration** и вновь запустить IEC-приложение.

ОБРАЩЕНИЕ В СЛУЖБУ ТЕХНИЧЕСКОЙ ПОДДЕРЖКИ

Для обращения в техническую поддержку Пользователю необходимо сформировать запрос на сайте технической поддержки: <https://support.prosoftsystems.ru>, либо отправить письмо по электронной почте: support@prosoftsystems.ru. В первом случае требуется предварительная регистрация.

Обращение обязательно должно содержать следующие сведения:

- подробное описание сложившейся ситуации;
- наименование объекта и его месторасположение;
- наименование системы автоматизации;
- модель ПЛК;
- серийный номер ПЛК;
- версия пакета обновления для среды разработки Astra.IDE;
- версия СПО контроллера;
- архив с лог-файлами (см. документ «Astra.IDE User Guide DPA 302. Раздел «Журнал событий»);
- архив с лог-файлами, включающими в себя период времени, когда произошел отказ;
- дата и время возникновения отказа. А также периодичность и устойчивость повторения подобных отказов в случае, если такая информация имеется.

Желательно прислать проект для Astra.IDE, так как это может значительно упростить и ускорить процесс поиска причины отказа.

Для того, чтобы узнать, как получить необходимую информацию (сведений о версии Astra.IDE, версии СПО и так далее), ознакомьтесь с содержанием документа «Astra.IDE User Guide DPA 302».

ПРИЛОЖЕНИЕ А

Настройка конфигурационного файла ServerConfig.xml

Конфигурационный файл находится в каталоге `/etc/OpcUA/ServerConfig.xml` и содержит некоторые настройки системы, доступные пользователю для редактирования. Не рекомендуется изменять предустановленные значения параметров в файле.

Таблица А.1 – Параметры конфигурационного файла

| Параметр | Описание |
|--|---|
| MaxDataQueueSize | <p>Максимальный размер очереди сообщений. Количество записанных переменных, с интервалом опроса <code>SamplingRate</code>.</p> <p>Сервер накапливает сообщения с интервалом <code>SamplingRate</code> до количества, указанного в клиенте, но не более настройки:</p> <pre><!--Maximum size of monitored item data queues.--> <MaxDataQueueSize>1000</MaxDataQueueSize></pre> |
| MinPublishingInterval / MaxPublishingInterval | <p>Минимальный/максимальный publishing интервал.</p> <p>Интервал публикации определяет частоту, с которой сервер проверяет наличие пакетов уведомлений для подписки, отправляемой обратно клиенту, т.е. интервал, с которым клиент выкачивает очередь записанных переменных.</p> <p>Например: <code>SamplingRate</code> – 50 мс (интервал, с которым сервер опрашивает переменные в приложении), <code>DataQueueSize</code> (настройка на клиенте) – 10, <code>PublishingInterval</code> – 1000 мс (интервал, с которым клиент получает “публикации”). Таким образом, клиент раз в секунду будет получать пачку состоящую максимально из 10 значений переменной (записанных с интервалом в 50 мс).</p> <p>Данная настройка ограничивает максимальный и минимальный <code>PublishingInterval</code>, который может выставить клиент (0 – без ограничения (Max), для значения Min - не имеет смысла ставить меньше, чем наименьший используемый <code>sampling</code> интервал)</p> <pre><!--Minimum publishing interval in milliseconds the server allows--> <MinPublishingInterval>50</MinPublishingInterval> <!--Maximum publishing interval in milliseconds the server allows. Default value 0 is no limitation--> <MaxPublishingInterval>0</MaxPublishingInterval></pre> |

| Параметр | Описание |
|---------------------------------|--|
| MinSupportedSamplingRate | <p>Минимальный sampling интервал. MinSupportedSampleRate задает значение переменной при запросе конфигурации сервера клиентом, а реальным сэмлированием сервера управляет набор значений SamplingRate, где первое значение определяет минимальный период сэмлирования сервера, который также может автоматически увеличиваться в зависимости от нагрузки.</p> <pre> <!--Minimum sample interval supported by the server--> <!-- avl was 0, and 0 in fact means 1, avl if 1 is set then we do not sleep at all. that is 1 means 0 --> <MinSupportedSampleRate>50</MinSupportedSampleRate> <!--Settings for the sampling engine.--> <AvailableSamplingRates> <!-- <SamplingRate>0</SamplingRate> --> <SamplingRate>50</SamplingRate> <SamplingRate>100</SamplingRate> <SamplingRate>250</SamplingRate> <SamplingRate>500</SamplingRate> <SamplingRate>1000</SamplingRate> <SamplingRate>2000</SamplingRate> <SamplingRate>5000</SamplingRate> <SamplingRate>10000</SamplingRate> </AvailableSamplingRates> </pre> <p>Установка интервала опроса не гарантирует обновление данных в указанный период, так как сервер автоматически подстраивает время (с шагом равным интервалу) исходя из текущей загрузки процессора</p> |