Программа OPC104 .

Руководство пользователя.

Общие положения

Программа OPC104 предназначена для использования, как в составе SCADA-систем, так и в качестве самостоятельного продукта. OPC104 позволяет решать следующие задачи:

1. Подключения OPC серверов и предоставления доступа к их данным по протоколам МЭК 60870-5-101/104;
2. Подключения удаленных источников данных по протоколам МЭК 60870-5-101/104 и предоставления доступа к данным этих УИД по протоколу OPC;
3. Передачи УИД команд управления и отображения результатов выполнения этих команд в виде тэгов OPC;
4. Подключения к удаленным приемникам данных по протоколам МЭК 60870-5-101/104.

Терминология.

Узел - экземпляр программного модуля, осуществляющий обмен данными с удаленным источником данных посредством протокола МЭК 60870. **Узел** является структурной единицей программы OPC104.

Peer - Отвечающая сторона, с которой **узел** обменивается данными.

Модель данных.

В рамках программы нет деления на узлы-получатели и узлы-источники данных. Любой узел может выступать одновременно и в качестве получателя и в качестве источника данных. Настраивается только **РОЛЬ УЗЛА ПРИ УСТАНОВКЕ СОЕДИНЕНИЯ** (активная или пассивная).

В случае МЭК 60870-104 активным будет узел, устанавливающий соединение.

В случае МЭК 60870-101 активным будет **Primary** узел (режим **Unbalanced**).

Для **Balanced** режима обмена по протоколу МЭК 60870-101, когда обе стороны обмена считаются равноранговыми, настройка **Primary** задает значение бита **DIR** в передаваемых кадрах. Если в **Balanced** режиме флаг **Primary** установлен, бит **DIR** в передаваемых кадрах устанавливается равным 1.

Структурная схема программы представлена на рисунке 1.

С точки зрения модели данных, узлы, работающие по протоколу МЭК 60870-101 и МЭК 60870-104 схожи и поэтому на структурной схеме (рис 1) они представлены одинаковыми блоками.

Любой узел, относящийся к IEC 60870 протоколу, имеет 3 внутренних набора ссылок на данные, хранящиеся в глобальном пространстве данных OPC104:

запрашиваемый набор (в дальнейшем **Requested Set**)

предоставляемый набор (**Serviced Set**)

набор команд (**Command Set**).

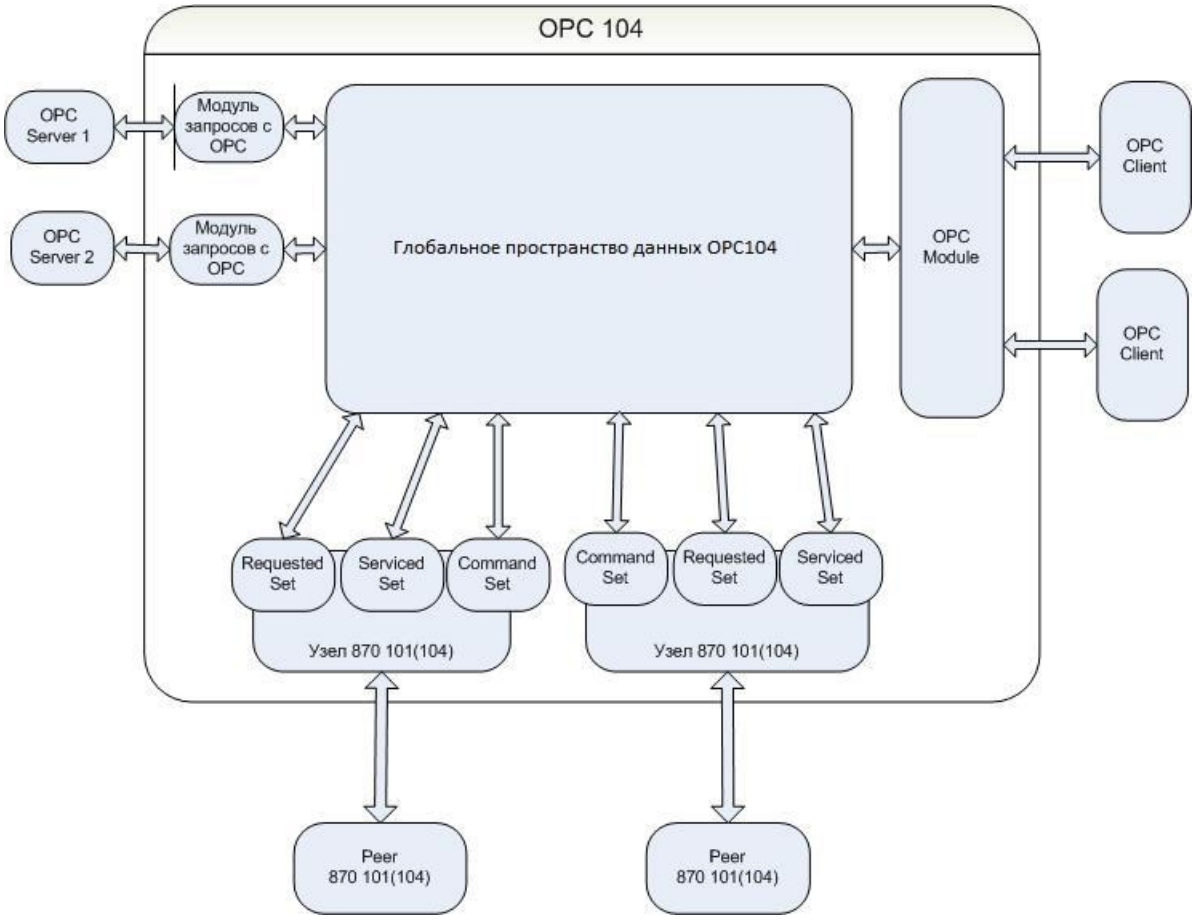


Рис 1.

Как видно из рисунка 2, каждому сигналу в наборе **Serviced Set** присваивается индивидуальный адрес сигнала и индивидуальный **TYPE_ID** , который будет использоваться программой для кодировки сигнала. Таким образом, можно настроить программу, чтобы она отдавала индивидуальный набор сигналов клиенту A и другой набор сигналов клиенту C (D,E и.т.д.).

Применение индивидуальных **Serviced Set** имеет несколько преимуществ по сравнению с глобальным **Serviced Set** (модель данных применявшаяся в версии 1).

- Сокращается объем данных отдаваемых как по запросам General Interrogation, так и спонтанными посылками. Не передаются «лишние» данные, в которых получатель не заинтересован.
- Пользователь может индивидуально настроить область видимости данных для каждого получателя. Т.о. получатель видит только то, что ему позволено видеть посредством множества **Serviced Set**.
- Пользователь может индивидуально настроит типы данных предоставляемых получателю.

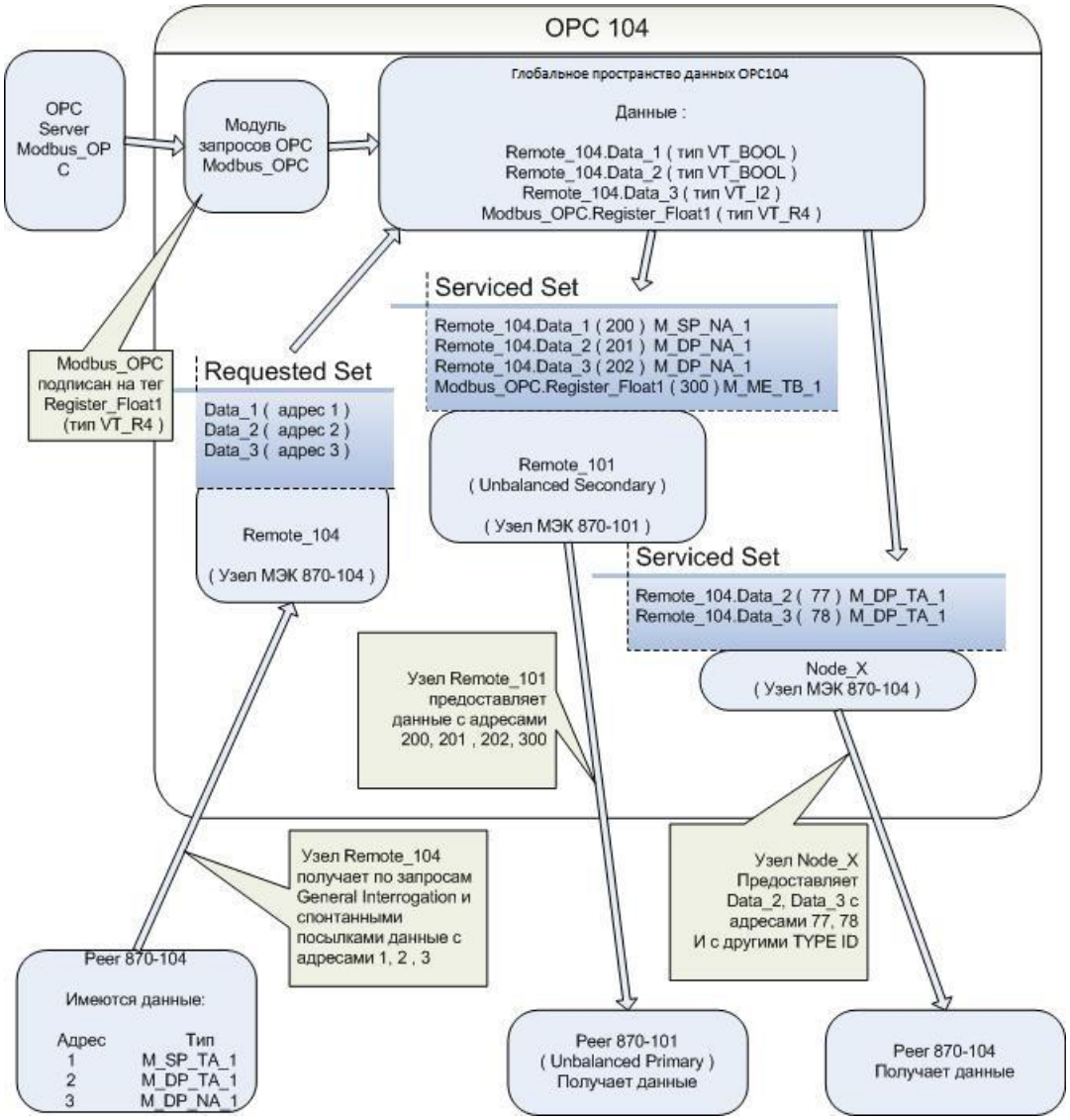


Рис 2.

Настройка Serviced Set.

Для настройки **Serviced Set,** щелкните правой кнопкой мыши по узлу, который вы хотите настраивать. Выберите пункт меню - "Выбрать для добавления серв. тегов".

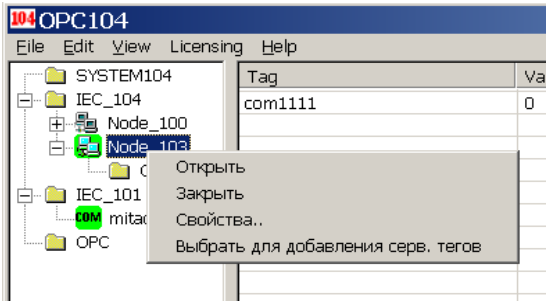


Рис 3.

После того как настраиваемый узел выбран, следует выделить нужный тег в таблице сигналов, и из контекстного меню выбрать пункт"Add to serviced".

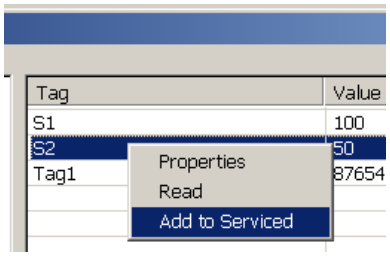


Рис 4.

Появляется диалог «Добавление к Serviced Set» (рис. 5) при появлении которого программа показывает текущий разрешенный адрес для данного сервера. Например, если Serviced Set уже имеет 3 ссылки на сигналы и для них заданы адреса 1, 2, 3, то в качестве текущего разрешенного адреса программа установит адрес 4.

Возможные проблемы:

- Если не был выбран узел, для которого определяется **Serviced Set**, диалог «Добавление к Serviced Set» не появится, а в строке-статусе программы появится сообщение о том, что не выбран узел для добавления сервисных сигналов.

- Если сигнал, который выбран для добавления, уже имеется в редактируемом Serviced Set, программа сообщит об этом в нижнем окне сообщений, диалог добавления сигнала в этом случае так же не появится .

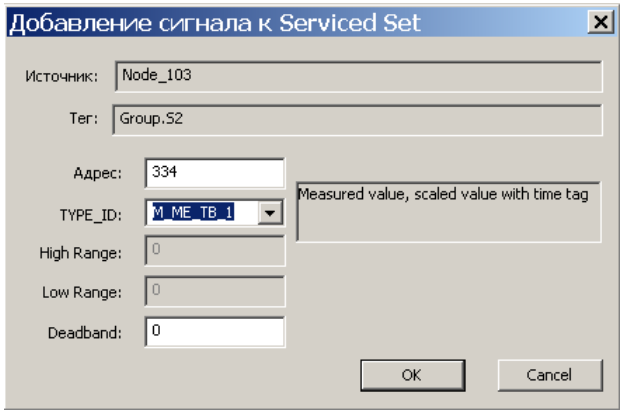


Рисунок 5.

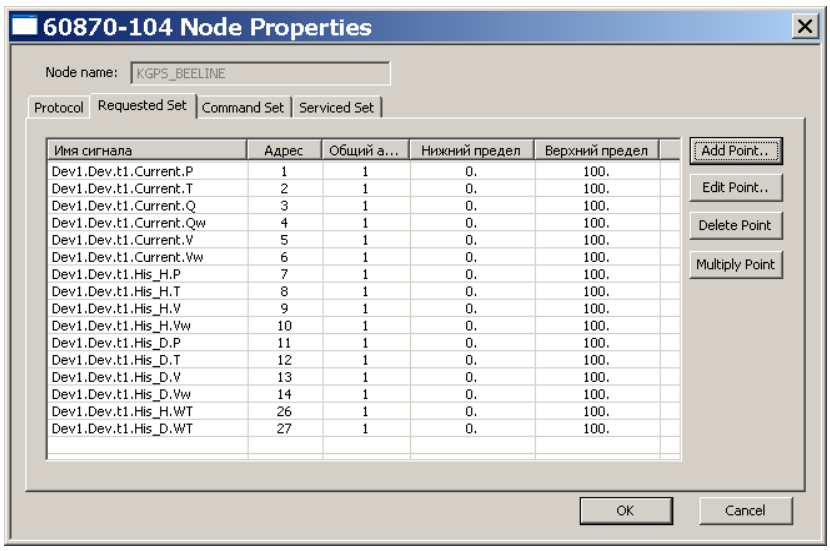
В появившемся диалоге имеется возможность задать Адрес, TYPE_ID и параметры High Range, Low Range , Dead band назначение которых будет объяснено позже. Если TYPE_ID будет оставлен пустым, программа сама будет подбирать его исходя из текущего типа сигнала. Например, для VT_BOOL, TYPE_ID будет выбран M_SP_TA_1 и.т.д.

Следует отметить, что узел может иметь в составе **Serviced Set** любые сигналы. Источниками сигналов могут служить как OPC теги, так и сигналы, получаемые посредством протокола МЭК 60870. Узел может иметь в наборе **Serviced Set** даже те сигналы, которые он получает **peer**-а!

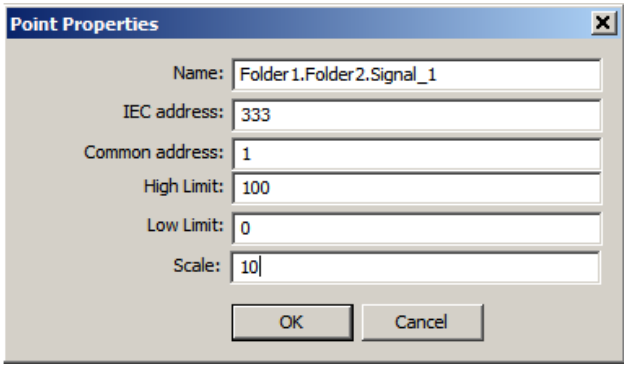
В этом случае **узел**, получив сигнал, зеркально отражает его обратно. Это свойство программы может быть использования для оценки пропускной способности каналов связи, а так же, для определения времени задержки обновления сигналов.

Настройка запрашиваемых данных (Requested Set)

В свойствах узла протокола IEC 104 перейдите к закладке «Requested Set».



Пользователь может добавлять / редактировать / удалять данные запрашиваемые у peer-а используя кнопки Add, Edit, Delete. Диалог добавления/редактирования запрашиваемого сигнала имеет следующий вид :



В данном примере пользователем добавляется сигнал который будет иметь имя Folder1.Folder2.Signal_1 и адрес используемый для запроса будет 1:333 (1 – common address, 333 – information object address) . Тип сигнала будет зависеть от того с каким типом данных отправит этот сигнал сервер.

Поля High Limit, Low Limit задают границы значений тегов для типов имеющих normalized значения (iec60870-5-101 п. 7.2.6.6, типы M_ME_NA_1, M_ME_TA_1, M_ME_ND_1, M_ME_TD_1). Для таких типов, в случае, когда заданы границы, значение вычисляется по формуле:

v = s * (hi - lo) / 65536 + (lo + hi) / 2 ;

где

s - переданное 16 битное значение со знаком, hi – верхняя граница, lo – нижняя граница.

В конфигурационных файлах iec104.xml, iec101.xml данные параметры имеют имена HIGH, LOW.

Для принимаемых типов имеющих scaled значения (iec60870-5-101 п. 7.2.6.7, типы M_ME_NB_1, M_ME_TB_1, M_ME_TE_1) масштаб задается с помощью параметра SCALE. При этом значение вычисляется по формуле:

V = s * scale;

где

s - переданное 16 битное значение со знаком, scale – значение параметра SCALE.

Отображение поля качество стандарта IEC 60870-101 на качество OPC задается следующим образом :

8й бит (invalid) -> bad quality (0)

5й бит (blocked) -> out of service quality (0x1C)

7й бит (topical) -> last usable quality (4)

1й бит (overflow) -> устанавливается качество bad (0) только для типов у которых возможно переполнение (например тип float).

Пользователю дается возможность задавать свое отображение качества iec на качество opc и обратно.

Для этого в файле OPC104.ini в секции [SETTINGS] надо указать параметр OVERRIDE_QUALITY

[SETTINGS]

OVERRIDE_QUALITY=1

В том же каталоге файловой системы, где располагается файл OPC104.exe создать файл quality_map.csv. Каждая отдельная строка файла quality_map.csv задает отображение качества. Первой позицией идет значение качества iec в десятичной кодировке. После запятой следует соответствующее качество OPC в десятичной кодировке. Ниже приведен пример содержимого quality_map.csv :

20,192

В данном примере OPC104 получив качество 0x14 (20 дес.) установит качество 192 (GOOD). С помощью таблицы, заданной в файле quality_map.csv, задается трансляция качества iec->opc и обратно opc->iec.

Взаимодействие программы OPC104 с OPC серверами.

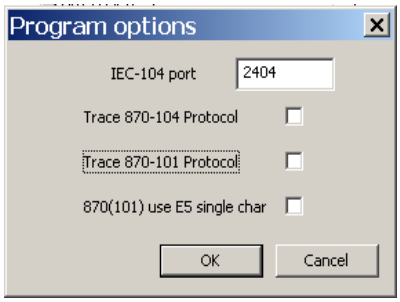
Во второй версии OPC104 по-другому построена работа с локальными **OPC** серверами. В отличие от предыдущей версии, программа не выполняет запросов адресного пространства **OPC** серверов при старте. Программа подписывается только на OPC-теги, которые надо отдавать дистанционным узлам (т.е. на которые имеются ссылки во множествах **Serviced Set**). При необходимости добавления новых сигналов, пользователь может инициировать запрос адресного пространства какого-либо OPC сервера. Такая схема работы имеет несколько преимуществ по сравнению со схемой, когда программа автоматически при старте выполняет запросы всех необходимых OPC серверов.

Существенно укорачивается время инициализации программы при старте. Некоторые OPC серверы могут иметь очень большое адресное пространство, для получения которого может уйти значительное время (от нескольких минут до нескольких часов!).
Программа не подключает OPC теги которые не нужны, что положительно сказывается на производительности.

Следует отметить, что в процессе работы программа выполняет постоянный контроль работоспособности подключенных OPC серверов, путем вызова метода GetState. Если вызов метода был неудачен, программа считает такой сервер «проблемным», прерывает соединение с ним и пытается заново «открыть» его. Так же важно, что контролируется каждый вызываемый метод каждого OPC сервера (например Read, Write, AddItems). И если истекает тайм-аут ожидания выполнения метода – программа информирует об этом пользователя записью (таймаут + имя метода + имя OPC сервера) в лог файл и в окно сообщений, и так же пытается заново открыть данный OPC сервер. Данные меры существенно повышают устойчивость и «живучесть» программы.

Общие настройки программы.

Диалог общих настроек программы представлен на рисунке 6



У пользователя имеется возможность задать :

- TCP порт прослушивания (порты активного соединения по умолчанию имеют значение 2404, но могут настраиваться индивидуально).

- Выполнять или нет трассировку протокола МЭК 60870 104 или МЭК 60870 101.

- Использовать ли единичный символ (0xE5) при ответах. Данная опция относится только к протоколу МЭК 60870-101 для небалансного вторичного метода обмена данными.

Трассировка позволяет получить дамп обмена по выбранному протоколу. Файлы трассировки имеют имя trace104.log, trace101.log и располагаются в каталоге ./OPC104.LOG.

Трассировка может существенно снижать производительность программы. Поэтому НЕ следует забывать **выключать** флаги трассировки, после того как Вы получили срез протокола в файлах трассировки.

Примерное содержание файла трассировки для протокола МЭК 60870 101 приведено ниже:

09-04-17 08:59:51 COM2 <- : 10 49 01 4A 16

09-04-17 08:59:51 COM2 -> : 10 0B 01 0C 16

Из первой строки следует, что программа получила по коммуникационному порту COM2 байты 10 49 01 4A 16 (данные представляются в шестнадцатеричном виде). Вторая строка информирует о том, что программа отправила байты 10 0B 01 0c 16, так же используя порт COM2.

Начиная с версии 2.91, программа может работать в режиме сервиса. Чтобы настроить программу для работы в режиме windows-сервиса следует в файле OPC104.ini установить настройку SERVICE в 1. Пример –

[SETTINGS]

SERVICE=1

Для регистрации программы в реестре в режиме сервиса следует выполнить:

$ OPC104.exe –install

(опцию SERVICE установить в 1)

Для де-регистрации программы в реестре в режиме сервиса следует выполнить:

$ OPC104.exe –uninstall

Для регистрации программы в реестре в интерактивном режиме следует выполнить:

$ OPC104.exe –regserver

(опцию SERVICE установить в 0)

Для де-регистрации программы работавшей в интерактивном режиме следует выполнить:

$ OPC104.exe -unregserver

Настройка OPC серверов

Для каждого подключаемого OPC сервера имеется возможность задавать индивидуальные параметры подключения Update Rate (интервал обновления изменившихся тегов) и Dead Band (полоса нечувствительности). Данные параметры определены стандартом OPC и предназначаются для уменьшения использования ресурсов. Они определяют то, как часто происходят уведомления клиента о смене параметров тегов. Update Rate задает интервал времени в миллисекундах по истечении которого OPC Server должен известить OPC клиента об изменениях параметров тегов (значение, время, качество), если эти изменения имели место в прошедший интервал. Dead Band определяет зону нечувствительности . Если разница между предыдущим значением тега и его текущим значением не превышает значения, определяемого параметром Dead Band, уведомление клиента не происходит. Параметр Dead Band задается в процентах.

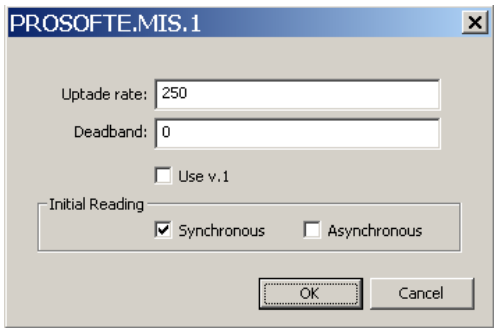


Рисунок 6.

Кроме стандартных параметров подключения определенных в стандарте OPC существуют дополнительные параметры OPC сервера. Ниже приведены их наименования и назначение.

Use V1. Если известно, что данный OPC сервер работает лучше, когда в качестве механизма уведомлений используется интерфейс IAdviseSink определенный в стандарте OPC v 1, следует установить этот флаг.

Группа параметров Initial Reading

OPC сервер, выполненный в соответствии с рекомендациями приведенными в стандарте OPC, после подключения тегов (выполнения метода AddItems) должен уведомить клиента о текущих значениях тегов используя функции обратного вызова. Если же какой-либо OPC сервер не следует данным рекомендациям, то для получения начальных значений тегов следует установить флажки Initial Reading Synch, или Initial Reading Asynch. В этом случае программа OPC104 выполнит синхронное или асинхронное чтение подключенных тегов.

Обновление списка запрашиваемых OPC тегов.

В случае, когда сигнал, источником которого является OPC сервер, не подключен, выберите пункт меню “Edit” -> "Refresh OPC list". В появившемся диалоге «Список доступных OPC серверов» (рис. 7) имеется возможность получить список доступных на данном компьютере OPC серверов. Для получения списка нажмите на кнопку «Получить список серверов». Получение списка OPC серверов происходит с помощью стандартного компонента OPCEnum.exe. После получения списка установите курсор на одном из OPC серверов. При этом выбранное имя скопируется в строку редактирования «Выбранный OPC сервер». Если по каким-то причинам OPCEnum работает не корректно, то имеется возможность вручную указать имя OPC сервера (его ProgID) просто задав его в строке редактирования «Выбранный OPC сервер» вручную.

После того как строка «Выбранный OPC сервер» содержит имя нужного вам OPC сервера, нажмите кнопку "Получить теги". Будет выполнена операция получения адресного пространства тегов данного OPC сервера(browse), иерархия тегов будет отражена в дереве узлов (левая часть окна) программы OPC104 и вы сможете добавить теги к набору Serviced Points какого либо узла. При следующем старте программа подключит данные теги автоматически. Набор Serviced Points хранится в файле iec104.xml (iec101.xml) для каждого узла отдельно в XML элементе <SERVED_POINTS>.

В случае перехода на новую версию, программа не обнаруживает раздела <SERVED_POINTS> и переходит к процессу инициализации этого раздела. Для этого она просматривает файл

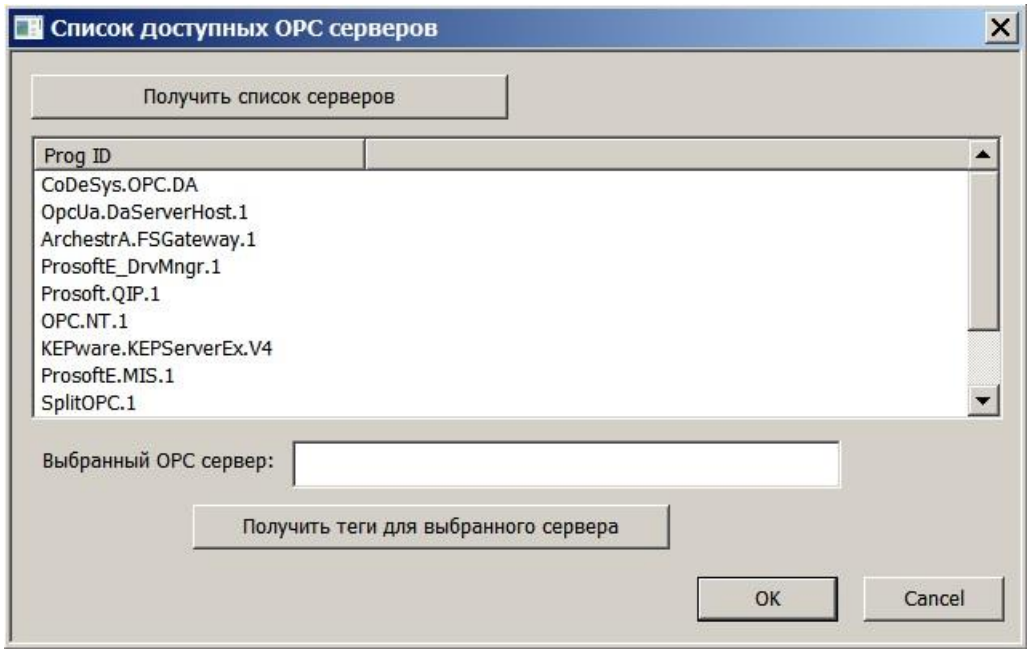


Рисунок 7.

opcpoints.xml оставшийся от версии 1 и копирует сигналы, перечисленные в opcpoints.xml, для каждого узла в соответствующий элемент <SERVED_POINTS>. Впоследствии пользователь может добавлять/удалять/редактировать этот набор индивидуально для каждого узла.

Редактирование свойств узлов.

Редактирование сигналов/команд запрашиваемых и предоставляемых доступно в диалоге "Свойства узла" (101 и 104). Следует отметить, что диалоги настройки протоколов 101 и 104 имеют много общего и различаются только в части настройки среды передачи сигналов. Поэтому вначале будет разобрана общая часть настроек, а затем специфичные параметры для каждой среды передачи сигналов.

Диалог настроек узла, работающего по протоколу IEC 60870-104, представлен на рисунке 8. К общим параметрам для 104 и 101 относят наборы сигналов **Requested Set** , **Command Set**, **Serviced Set**. В нижней части диалога располагается элемент управления «блокнот», с тремя закладками. В каждой из закладок содержится таблица, содержащая один из наборов. Параметры, задающие длину полей, и параметр Common Address также являются общими для протоколов МЭК 60870.

Common address (bytes) – задает длину поля common address . Допустимые значения 1,2

Object address (bytes) – задает длину поля Information Address. Допустимые значения 1,2,3

Cause of transmission (bytes) – задает длину поля Cause of Transmission. Допустимые значения 1,2

Common address - задает общий адрес.

Использ.локальное время – Определяет способ отображения меток времени. Если отметка выключена, то метки времени отображаются в базисе UTC. Если отметка установлена, то эти метки приводятся к локальному базису времени (например, UTC+5).

Интервал общ.опроса (сек) – Интервал времени в секундах, по истечении которого источнику данных подается команда общего опроса General Interrogation.

Active Connection – Определяет роль узла при установке соединения. В случае 101-го протокола соответствующий параметр называется **Primary**

Buffer Reports Определяет следует ли буферировать спонтанные уведомления. Если флаг **Buffer Reports** установлен, то здесь же можно задать интервал буферирования в миллисекундах. Обычно этот параметр имеет значение в пределах 100-200 миллисекунд. Буферирование спонтанных уведомлений существенно увеличивает эффективность использования канала передачи данных, т.к. при буферизации в одном PDU высылается сразу же несколько значений сигналов. На практике, для протокола МЭК 870-101, буферирование как минимум в 10 раз увеличивает количество сигналов передаваемых без накопления очереди. Исходя из этого, можно рекомендовать ВСЕГДА устанавливать флаг Buffer Reports. Следует так же отметить, что буферирование применяется только к измеряемым значениям (M_ME_***) к

дискретным сигналам, имеющим тип M_SP_** , M_DP_** , буферирование не применяется никогда.

Алгоритм буферирования спонтанных уведомлений следующий: при возникновении события смены параметров сигнала, программа не высылает PDU сразу, а чуть придерживает его на время **не превышающее** интервал буферирования. Если, в течении этого интервала, возникают события смены параметров сигналов имеющие такой же TYPE ID, программа добавляет их в текущий буфер. При заполнении буфера (максимум 249 байт) программа отсылает его безотносительно того истек ли интервал буферирования, или нет. Поэтому данный интервал следует рассматривать как **максимальную задержку**, т.к. данные могут отсылаться и раньше, чем по истечении данного интервала.

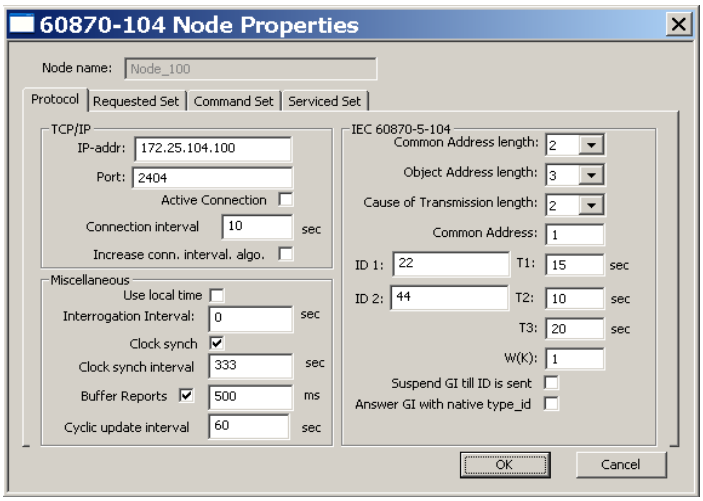


Рисунок 8.

К специфическим параметрам для МЭК 60870-104 относятся параметры:

IP адрес и порт - Определяют сетевое имя (или ip-адрес или доменное имя узла) и порт TCP соединения. Индивидуально порт может задаваться только для активного узла. Для узлов которые принимают соединения (неактивны) прослушиваемый порт задается глобальным параметром, который можно изменить в общих свойствах программы.

Connection interval - Программа делает попытки соединения каждые 20 секунд. Если параметр Increase connection interval установлен , и попытка соединения была неудачна, то интервал увеличивается в 2 раза (40 сек) и т. д. до тех пор, пока данный интервал не достигнет 400 секунд. Более 400 секунд он не увеличивается. После удачного соединения этот интервал вновь устанавливается равным 20 сек. Сделано это из расчета, что если peer долго не

отвечает, то скорее всего, он в данный момент не работает и не следует слишком часто пробивать соединение для экономии трафика. Если же peer отвечал недавно, то возможно, что обрыв связи был вызван не выключением peer-а, а проблемами со связью и в этом случае следует чаще пробивать соединение. В случае **unbalanced secondary** режима данный параметр означает период времени в течении которого должна происходить какой-то обмен данных. Если за данный период не было обмена, то соединение считается утраченным. Если данный параметр задан равным нулю, то в **unbalanced secondary** режиме НЕ происходит проверки активности канала связи.

Increase connection interval - Параметр управляющий алгоритмом увеличения интервала соединения (см. выше) Если флажок сброшен, то интервал попыток соединения будет оставаться неизменным.

Макс.T1 (мсек)» – Тайм-аут посылки информационных или тестовых APDU. (Максимальное время ожидания подтверждения отправленных APDU).

«Макс.T2 (мсек)» – Тайм-аут для подтверждений в случае отсутствия информационных сообщений (T2 < T1). (Максимальное время ожидания получения W штук APDU. Если до истечения T2 не было получено W штук APDU, то отправляется подтверждение (формата-S).

«Макс.T3 (мсек)» – Тайм-аут для посылки тест-фреймов в случае длительного бездействия системы.

«Макс.W» – Количество полученных сообщений, при превышении которого источнику данных отправляется подтверждение (формата S).

«Cyclic update interval» - Интервал, заданный в секундах, по истечении которого отсылаются все данные из набора Serviced Set с причиной передачи cyclic/periodic (1). Если данный интервал равен нулю, то программа не делает циклического обновления данных.

Иногда, в случае работы в неактивном режиме, (соединение устанавливается к нам) когда мы не получаем никаких данных от peer-а, становиться нежелательной посылка команды General Interrogation. В этом случае задайте параметр INITIAL_INTERROGATION=”0” в файле iec104.xml. Например:

<SLAVE>

<PARAMS ACTIVE_CONNECT="0" INITIAL_INTERROGATION="0"

…>

Вид диалога настроек узла работающего по протоколу МЭК 60870 101 приведен на рисунке 9 и 10.

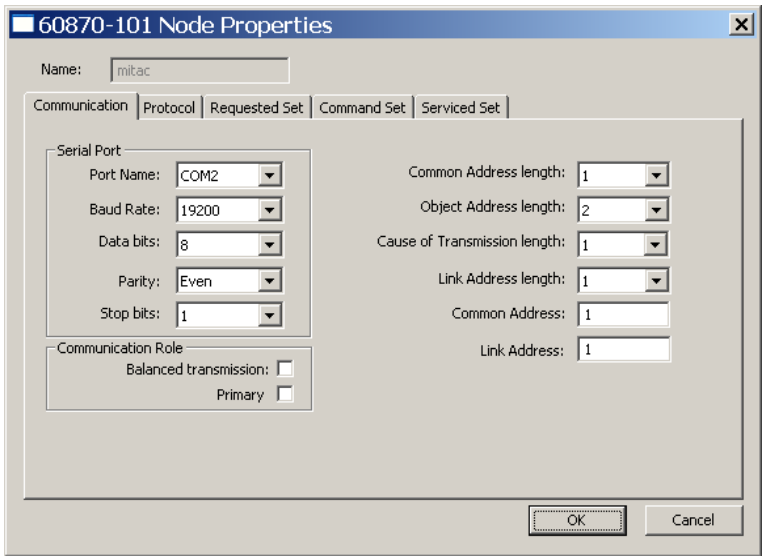


Рисунок 9.

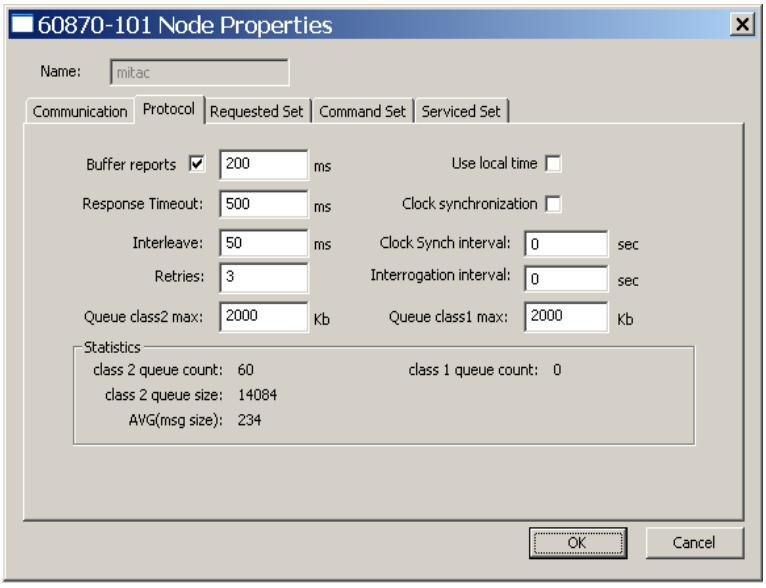


Рисунок 10.

Специфические параметры для протокола МЭК 60870 101

PortName	Выпадающий список с допустимыми именами последовательных портов (допускаются имена COM1, COM2, … COM9)
Baud Rate	Выпадающий список, содержащий допустимые значения скорости обмена данными с портом (по умолчанию – 38400)
Data Bits	Выпадающий список, содержащий допустимые значения количества битов данных (по умолчанию – 8)
Parity	Выпадающий список допустимых методов контроля четности (по умолчанию – нет)
Stop Bits	Выпадающий список, содержащий допустимые значения количества стоповых битов (по умолчанию – 1)

Link Address Length Задает длину поля Link Address. Допустимое значение 0, 1 или 2.

Link Address Значение поля LINK ADDRESS для процедур обмена канального уровня.

Balanced Определяет режим обмена канального уровня (балансный или небалансный)

Response Interval интервал времени задаваемый в миллисекундах называемый T0 на диаграммах перехода состояний.

Для небалансного **primary** – определяет интервал по истечении которого время программа считает, что ответ на запрос не пришел.

В unbalanced режиме поддерживается режим «несколько устройств разделяют один коммуникационный порт». Это применимо и для primary и для secondary устройств. Каждое из устройств в этом случае должно иметь уникальный link-адрес. Важно чтобы параметры COM-порта были одинаковы для всех устройств разделяющих общий канал связи. Если, по ошибке, параметры COM-порта не одинаковы, то фактически будут применяться параметры устройства указанного раньше в конфигурационном файле.

Порядок выбора параметра Response Interval.

На практике значение этого параметра определяется не только полосой пропускания канала, но и особенностями отвечающей стороны (peer-a). Согласно стандарту МЭК 60870-101 (п. 6.2.2.1) рекомендованы следующие значения T0 для небалансного режима обмена.

Baud Rate	100	600	1 200	9 600	19200	64 000
T0 (ms)	26 460	4 451	2 250	325	187	91.3

При часто случающихся таймаутах T0, программа OPC104 меняет цвет узла с зеленого на красный, сигнализируя о том, что, возможно, величина T0 задана слишком малой. Возможно, что в этом случае понадобится установить значение T0 на большее.

Interleave interval Значение этого параметра имеет различное применение применительно к балансному и небалансному методам обмена информацией.

Для небалансного **primary** – интервал циклического опроса устройства (приемлемые значения 4 – 1000 мс). **Следует отметить, что данный параметр СУЩЕСТВЕННО влияет на объем передаваемой информации, чем меньше интервал опроса, тем больший объем информации может получать программа.** Но не рекомендуется делать его меньше 4

	миллисекунд, т.к. между циклами обмена информацией должны быть паузы. Для небалансного **secondary** – определяет интервал выжидания после получения ошибочного пакета (например содержащего ошибочную контрольную сумму) в течении этого интервала программа игнорирует поступающие данные. Стандарт определяет этот интервал как интервал прохождения 33 бит. Так для скорости обмена 19200 бод этот параметр можно установить равным 2 миллисекундам.
Retries	Для небалансного **primary** – определяет количество повторов посылки запроса, исчерпав которое, программа переходит в начальное (initial) состояние и заново устанавливает соединение.

Добавление / удаление узлов.

Для добавления узла щелкните правой кнопкой мыши по корневому каталогу в древовидном списке в левой части главного окна программы. IEC_101 для добавления узла работающего по протоколу МЭК 60870 -101 или по каталогу IEC_104 для добавления узла работающего по протоколу МЭК 60870 – 104. В появившемся контекстном меню выберите пункт «Новый узел». В появившемся диалоге свойств узла задайте желаемые параметры узла. Следует отметить, что при добавлении узла диалог свойств не позволяет редактировать наборы данных (Requested Set, Command Set, Serviced Set). Эта возможность появляется после создания узла и появления его в древовидном списке.

Для удаления узла щелкните правой кнопкой мыши по иконке узла в древовидном списке и выберите пункт меню «Удалить». После подтверждения Вами вашего намерения удалить узел программа удалит узел и все настройки связанные с ним.

Так же добавление / удаление / редактирование свойств узла возможно посредством системных данных, описанных ниже в разделе **Системные данные**.

Индикация узлов

Посредством индикации программа отображает текущее состояние узла. Для узлов, работающих по протоколу МЭК 60870 101, применяется следующая индикация:

Рабочее состояние.

Unbalanced secondary – переполнение очереди сообщений, Unbalanced primary – большое количество таймаутов за последние 30 секунд.

Порт закрыт.

Порт открыт, но связь в соответствии с протоколом не установлена.

Очереди сообщений.

OPC104 использует следующую стратегию при высылке сообщений:

Имеются две очереди сообщений :

Очередь класса 1 – для сигналов имеющих следующие TYPE ID :

M_SP_NA_1, M_SP_TA_1, M_DP_NA_1, M_DP_TA_1, M_ST_NA_1, M_ST_TA_1, C_SC_NA_1, C_DC_NA_1, C_RC_NA_1, C_SE_NA_1, C_SE_NB_1, C_SE_NC_1, C_BO_NA_1, C_SC_TA_1, C_DC_TA_1, C_RC_TA_1, C_SE_TA_1, C_SE_TB_1, C_SE_TC_1, C_BO_TA_1, M_EI_NA_1, C_IC_NA_1, C_CI_NA_1, C_RD_NA_1, C_CS_NA_1, C_RP_NA_1, C_CD_NA_1.

Очередь класса 2 – все остальные TYPE ID.

Очередь класса 1 имеет более высокий приоритет. На запрос «требуются данные класса 2» программа может ответить данными класса 1, если очередь класса 1 не пуста. Чаще всего переполнение очереди происходит именно для класса 2. Происходит это по причине того, что эта очередь менее приоритетная, а так же и потому, что измеряемые значения (ТИ) чаще меняют свое значение. В случае переполнения очереди сообщений, программа индицирует возникновение этого состояния сменой иконки узла и делает запись в лог-файл. Следует отметить, что даже в случае переполнения программа **не отбрасывает** сообщение, которое вызвало переполнение, а ждет когда размер очереди вернется к приемлемому, и тогда помещает вновь возникшее сообщение в очередь. Верхнюю границу очереди сообщений можно задать в диалоге «Свойства узла» рассмотренного выше. По умолчанию размер очередей равен 2 мегабайтам. Текущее состояние очередей сообщений можно посмотреть в диалоге «Свойства узла» на вкладке «Protocol» . В случае включенного режима буферизации, очередь класса 2 содержит сообщения (PDU) которые, возможно, содержат в себе несколько уведомлений о смене состояния данных.

Нормальный режим работы программы соответствует размерам очередей близких к нулю.

Рост размера очереди сообщений свидетельствует о том, что пропускная способность канала передачи данных не соответствует объему информации, которую необходимо передать. При возникновении проблем, связанных с ростом размера очереди сообщений, в первую очередь следует рассмотреть параметр отвечающий за частоту опроса **на запрашивающей стороне** т.е. стороне получающей данные. Для OPC104 это будет параметр **Interleave Interval** . Иными словами - увеличить частоту опроса. Большое значение параметра «Верхняя граница размера очереди сообщений» поможет справляться только с пиковыми информационными нагрузками, в течении которых размер очереди будет расти, а после снижения нагрузки – уменьшаться. Если же средняя информационная нагрузка превышает возможности канала передачи данных, следует рассмотреть способы снижения объема передаваемых данных. Если источником передаваемых данных является OPC сервер, то можно рассмотреть вариант увеличения интервала времени **Update Rate** для этого OPC сервера. **Буферизация** отправляемых сообщений так же существенно увеличивает количество передаваемой информации передаваемой в единицу времени.

Команды управления.

Команды управления по своей структуре в рамках программы делятся на простые (direct) и комплексные (Select Before Operate). В диалоге Command Properties тип команды можно задать сняв или установив флажок Simple Command.

Добавление простой команды.

Откройте диалог свойств узла, выберите вкладку блокнота **Command Set.** Нажмите кнопку **Add**. В появившемся диалоге

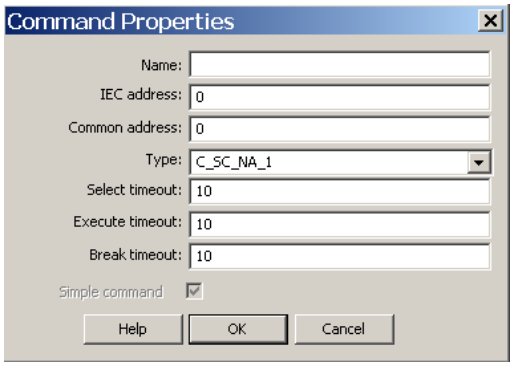


Рис 10.

Задайте имя команды ее адрес и тип. Так же имеется возможность задать интервалы команды. Для простой команды имеет смысл только параметр Execute timeout.

На данный момент реализовано только инициирование выполнение команды, но не исполнение. Реализовано инициирование следующих типов команд:

C_SC_NA_1 - однопозиционная команда

C_DC_NA_1 - двухпозиционная команда

C_RC_NA_1 - команда пошагового регулирования

C_SE_NA_1 - команда уставки, нормализованное значение

C_SE_NB_1 - команда уставки, масштабированное значение

C_SE_NC_1 - команда уставки, короткий формат с плавающей запятой

C_BO_NA_1 - строка из 32 бит

Логика работы простых команд.

Простые команды могут применяться только в случае прямого управления устройством. Простые команды **не** могут применяться в случае когда требуется функциональность выбор устройства перед управлением (select before operate или **SBO**).

Простые команды делятся на активные и обслуживаемые.

Активные команды: программа пересылает команду по протоколу IEC-101(104) и получает ответ о выполнении команды.

Обслуживаемые команды: программа получает запрос на выполнение команды по протоколу IEC-101(104) пытается выполнить команду путем записи заданного в команде значения в OPC-тег либо передав его дальше по протоколу IEC-101(104) по цепочке.

Выполнение активных команд: после получения запроса (например произошло выполнение операции Write по OPC-протоколу), OPC104 отправляет PDU содержащее код команды, являющийся атрибутом команды, и собственно команду (например 0 или 1 в случае C_SC_NA_1). Одновременно запускается таймер контроля выполнения команды. Если от дистанционного устройства приходит ответ о выполнении команды до момента срабатывания таймера, данное, представляющее команду, принимает значение, полученное в ответе от peer-а, устанавливает качество GOOD и время соответствующее времени ответа. Если же раньше сработает контрольный таймер – программа выставляет качество BAD данной команде.

Выполнение обслуживаемых команд: после получения запроса на выполнение команды (например принято PDU типа C_SC_NA_1) программа находит OPC-тег заданный в настройках данной обслуживаемой команды и пытается выполнить запись в этот тег. В случае успешной записи, программа отвечает положительным подтверждением команды. В случае неуспешного выполнения команды программа отвечает отрицательным кодом подтверждения команды. Неуспешное выполнение команды может произойти например потому что по заданному в PDU адресу не было найдено обслуживаемой команды, либо не существует OPC-тега заданного в настройках команды в поле ASSOCIATED_DATA.

Пример задания команды в файле iec104.xml (iec101.xml)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<NODES>

 <SLAVES>

   <SLAVE>

     <PARAMS ACTIVE_CONNECT="1" ASDU_ADDRESS_BYTES="2" BUFFER_INTERVAL="500"
BUFFER_REPORTS="1" CLOCK_SYNCH="0" CLOCK_SYNCH_CHECK="333" CONNECT_INTERVAL="10"
COT_BYTES="2" CYCLIC_UPDATE_INTERVAL="0" GI_NATIVE="0" ID1="0" ID2="0"
INCREASE_CONNECT_INTERVAL="0" INITIAL_INTERROGATION="1" INTERROGATION_CHECK="0"
IP_ADDRESS="172.29.22.103" MAX_T1="1" MAX_T2="10" MAX_T3="20" MAX_W="10"
NAME="Node_104" OBJ_ADDRESS_BYTES="3" PORT="2404" STATION_ADDRESS="7" SUSPEND_GI="0"
USE_LOCAL_TIME="0"/>

      <POINTS>

      </POINTS>

     <SERVED_POINTS>

    </SERVED_POINTS>

     <COMMANDS>

            <simple_2 ACTIVE="0" ADDRESS="2"
           ASSOCIATED_DATA="SUPERSOFT.OPC_SERVER.1.Group.Tag1" BREAK_TO="20"
           COMMON_ADDRESS="1" EXEC_TO="20" SELECT_TO="20" SIMPLE="0" SOURCE="
           SUPERSOFT.OPC_SERVER.1 " SOURCE_TYPE="OPC" TYPE_ID="C_SC_NA_1"/>
```

В данном примере задана команда с именем simple_2. В древовидной структуре OPC-тегов тег представляющий данную команду будет иметь вид : IEC_104.Node_104.simple_2.

TYPE_ID=”C_SC_NA_1” - задан тип команды

ACTIVE=0 – программа **отвечает** на пришедшие запросы типа C_SC_NA_1.

ASSOCIATED_DATA=”SUPERSOFT.OPC_SERVER.1.Group.Tag1” – полный идентификатор тега. При обработки полученной команды будет предпринята попытка записи в данный тег.

COMMON_ADDRESS, ADDRESS – Адрес команды.

SOURCE=”SUPERSOFT.OPC_SERVER.1” – Имя источника данных. В данном случае это имя OPC-сервера который содержит в себе тег Group.Tag1.

SOURCE_TYPE=”OPC” - Указывает на то что параметры SOURCE содержит в себе имя OPC-сервера.

BREAK_TO, SELECT_TO, EXEC_TO – таймауты выполнения команды.

Команды select before operate.

Поддержка команд типа select before operate выполнена следующим образом : в настройках программы задаются параметры - адрес команды, select таймаут в секундах, execute таймаут в секундах, break таймаут в секундах и тип команды.

Секция команд при этом может выглядеть следующим образом :

```
<NODES>
  <SLAVES>
    <SLAVE>
      <PARAMS ACTIVE_CONNECT="1" ASDU_ADDRESS_BYTES="2" BUFFER_INTERVAL="500"
BUFFER_REPORTS="1"  CLOCK_SYNCH="0" CLOCK_SYNCH_CHECK="333" CONNECT_INTERVAL="10"
COT_BYTES="2"   CYCLIC_UPDATE_INTERVAL="0" GI_NATIVE="1" ID1="22" ID2="44"
INCREASE_CONNECT_INTERVAL="0"      INTERROGATION_CHECK="0" IP_ADDRESS="10.10.10.10"
MAX_T1="15" MAX_T2="10" MAX_T3="20"      MAX_W="10" NAME="IEC104_NODE"
OBJ_ADDRESS_BYTES="3" PORT="2404" STATION_ADDRESS="1"    SUSPEND_GI="0"
USE_LOCAL_TIME="0"/>
  <COMMANDS>
    <complex_cmd    ADDRESS="205"   BREAK_TO="10"  COMMON_ADDRESS="1"   EXEC_TO="10"
SELECT_TO="10"  SIMPLE="0"    TYPE_ID="C_SC_NA_1"/>
  </COMMANDS>
```

Программа создает каталог под узлом, для которого задана команда. Для приведенного выше примера это будет:

IEC104_NODE

+-- complex_cmd

Внутри этого каталога будут созданы теги

select
select_state
break
break_state
execute
execute_state
reset

Назначение каждого тега приведено ниже:

- select - запись в этот тег приводит к посылке команды select

Условия выполнения этой команды: **select_state** должен быть в состоянии **sel_state_initial**. Иначе программа игнорирует запись в этот тег.

select_state - отображает текущее состояние, возможные значения :

sel_state_initial - изначальное состояние select

sel_state_sel_sent - команда select послана

sel_state_sel_confirmed_ok - команда select подтверждена

sel_state_sel_denied - пришел отрицательный ответ на команду select

sel_state_TO - произошел тайм аут выполнения команды select

- execute - запись в этот тег приводит к посылке команды execute . Тип данного зависит от типа команды, возможны типы – bool , int , float.

Условия выполнения этой команды **execute_state** должен быть в состоянии **exe_state_initial, select_state** в состоянии **sel_state_sel_confirmed_ok**. Иначе программа игнорирует запись в этот тег.

- execute_state - отображает текущее состояние, возможные значения :

exe_state_initial - изначальное состояние execute

exe_state_exe_sent - команда execute послана

exe_state_exe_confirmed_ok - команда execute подтверждена положительно

exe_state_exe_denied - пришел отрицательный ответ на команду execute

exe_state_TO - произошел тайм аут выполнения команды execute

- break - запись в этот тег приводит к отмене предварительно посланной перед этим команды select. Условия выполнения этой команды - select_state должен быть в состоянии **sel_state_sel_sent**.

- break_state - отображает текущее состояние, возможные значения :

break_state_initial **-** изначальное состояние

break_state_break_sent **-** команда break послана

break_state_break_confirmed_ok – команда break подтверждена

break_state_break_denied **-** пришел отрицательный ответ на команду break

break_state_TO **-** произошел тайм-аут выполнения break.

- reset – Запись любого значения в этот тег приводит в начальное состояние объект – команду. Т.е. select_state = **sel_state_initial,** execute_state = **exe_state_initial**, break_state = **break_state_initial**.

Системные данные.

Системные данные сгруппированы в каталоге SYSTEM104. Данная группа данных позволяет добавлять/удалять/редактировать узлы, работающие по протоколу МЭК 60870-104 посредством OPC интерфейса. Данное свойство программы позволяет пользователю автоматизировать процесс настройки. Особенно это полезно в случаях, когда узлы имеют однотипный набор данных и этот набор данных достаточно большой. Так же следует отметить что в таком случае можно применять скрипты написанные на языке Perl и подобных языках для редактирования файлов iec101.xml , iec104.xml в которых собственно и хранятся все настройки. Но редактирования свойств узлов посредством OPC интерфейса намного гибче.

Для автоматизации процесса настройки следует написать скрипт для программы OPC-клиента, содержащий необходимые узлы и сигналы. Ниже приведено назначение и примеры использования этих данных.

Имеется 6 системных тегов:

SYSTEM104.Add, SYSTEM104.Delete, SYSTEM104.Edit SYSTEM104.AddPoint, SYSTEM104.AddCommand и SYSTEM104.Refresh. Все они имеют тип String.

- SYSTEM104.Add

Пример формата строки :

NAME=NewNode ACTIVE_CONNECT=1 ASDU_ADDRESS_BYTES=2 COT_BYTES=2 INTERROGATION_CHECK=3600 IP_ADDRESS=10.10.10.1 MAX_T1=15 MAX_T2=10 MAX_T3=20 MAX_W=10 OBJ_ADDRESS_BYTES=3 STATION_ADDRESS=1 CLOCK_SYNCH=0

Порядок следования параметров не важен. Параметры NAME, IP_ADDRESS обязательны. Значения по умолчанию для остальных параметров приведены в вышестоящем примере. Запись строки, формат которой приведен выше, побуждает программу создать узел с таким именем, взяв его настройки из параметров строки и сохранить данные настройки в файле iec104.xml. Если операция добавления была успешной, значение тега меняется на "Success" в противном случае "Failure"

- SYSTEM104.Delete - запись в этот тег строки содержащей имя узла, побуждает программу удалить узел с таким именем. Так же изменяется содержание файла iec104.xml. Удаляется раздел содержащий настройки данного узла. Значение тега в случае успеха : "Delete Node_100 : Success". В случае неуспеха "Delete Node_100 : Failure"

SYSTEM104.Edit Формат строки приемлемой для записи в этот тег :

NAME=Node_100 ACTIVE_CONNECT=1 ASDU_ADDRESS_BYTES=2 COT_BYTES=2 INTERROGATION_CHECK=3600 IP_ADDRESS=172.25.104.100 MAX_T1=15 MAX_T2=10 MAX_T3=30 MAX_W=100 OBJ_ADDRESS_BYTES=3 STATION_ADDRESS=1

Может отсутствовать любой параметр кроме параметра NAME. Параметр NAME должен присутствовать всегда. Он не обязательно должен быть первым. Параметр NAME определяет узел, свойства которого редактируются. Назначение остальных параметров совпадает с назначением параметров описанных выше в разделе «Настройки программы OPC104».

Пример

Для того чтобы сменить ip-адрес для узла Node_100 можно записать строку вида NAME=Node_100 IP_ADDRESS=10.10.10.100. В случае успеха значение тега будет

«Edit Node_100 : Success»

в обратном случае (например, программа не нашла Node_100)

Edit Node_100 : Failure

SYSTEM104.Refresh

Запись в этот тег строки содержащей имя узла (например "Node_100"), побуждает программу перечитать раздел POINTS из файла iec104.xml и добавить теги, отсутствующие до этого. Удаления тегов, которые не присутствуют в разделе POINTS, не происходит.

SYSTEM104.AddPoint

Пример формата строки:

HOST=Host_100 POINT=Group.Data10 ADDRESS=10 COMMON_ADDRESS=1 SERVER_ADDRESS=0

Параметры HOST, POINT, ADDRESS обязательны. Значения по умолчанию для параметров COMMON_ADDRESS и SERVER_ADDRESS указаны выше в примере.

Операция записи в инициализирует добавление нового данного в **Requested Set** с заданными параметрами.

SYSTEM104.AddCommand

Пример формата строки:

HOST=Node_100 SIMPLE=1 COMMAND_NAME=Group.Tag1 TYPE_ID=C_SC_NA_1 ADDRESS=1001 COMMON_ADDRESS=1

Параметры HOST, COMMAND_NAME, TYPE_ID, ADDRESS обязательны.

Значения по умолчанию для остальных тегов приведены в вышестоящем примере.

Запись в данный тег инициализирует добавление новой команды с заданными параметрами в набор **Command Set** выбранного узла. В случае SIMPLE=1 будет добавлен OPC-тег с именем заданным в параметре COMMAND_NAME. В случае SIMPLE=0 будет добавлен каталог содержащий в себе стандартные командные данные.

Допустимые значения параметра TYPE_ID :

C_SC_NA_1 : Single Command (команда типа On/Off)

C_DC_NA_1 : Double command (команда типа On/Off имеющая третье состояние)

C_RC_NA_1 : Regulating step command

Значение команды - байт. Допустимые значения команды: 1- на ступень ниже 2- на ступень выше

C_SE_NA_1 : Set-point command, normalized value

Значение команды - знаковое 16 битное целое + 8 битный квалификатор. Значения квалификатора от 1 до 63 зарезервированы. Их семантика определена стандартом.

C_SE_NB_1 : Set-point command, scaled value

То же что и **C_SE_NA_1**, но положение десятичной точки определяется дополнительным параметром. Например, для передачи значения 10.3 передается целое число 103. Дополнительный параметр, определяющий положение десятичной точки определяет, что число 103 следует интерпретировать как 10.3.

C_SE_NC_1 : Set-point command, short floating point number. Значение параметра - float (4 байта) + дополнительный 8ми битный квалификатор.

C_PROSOFT_TIME : Специализированная команда использующаяся для задания временных интервалов (расширение стандарта IEC 60870-5-101).

SYSTEM104.SaveOptions

Начиная с версии 2.3 OPC104, не делает сохранения настроек в xml файлах после каждой операции по изменению опций. Программа просто запоминает сам факт того, что настройки были изменены. Запись в тег **SYSTEM104.SaveOptions** значения = 1 приводит к тому, что программа "сбрасывает" на диск параметры, хранящиеся в оперативной памяти. Так же это происходит при завершении работы программы и при выборе пункта меню "Настройки" -> "Сохранить Настройки"

Пример удаленного взаимодействия посредством SplitOPC с OPC104 (добавление и использование новых узлов, тегов и команд)

В данном примере имя удаленного узла Linux_104

1) Добавляем новый узел.

Записываем в тег SYSTEM104.Add значение

NAME=Linux104 IP_ADDRESS=172.25.104.104 ACTIVE_CONNECT=1 ASDU_ADDRESS_BYTES=1 COT_BYTES=1 MAX_T1=30 MAX_T2=10 MAX_T3=300 OBJ_ADDRESS_BYTES=2

(Убеждаемся что значение стало Success)

2) Выполняем команду добавления данных - записываем в тег SYSTEM104.AddPoint значение

HOST=Linux104 POINT=Group.Data20 ADDRESS=20

(Убеждаемся что значение стало Linux104 add point Group.Data20 : Success)

3) Выполняем команду чтения этого тега

Linux104.Group.Data20

4) Выполняем добавление команды - записываем в тег SYSTEM104.AddCommand значение

HOST=Linux104 COMMAND_NAME=Commands.Command30 ADDRESS=30 TYPE_ID=C_SC_NA_1

Убеждаемся что значение тега **SYSTEM104.AddCommand** стала

Linux104 Add Command Commands.Command30: Success

5) Выполняем запись в тег SYSTEM104.SaveOptions

Динамическая идентификация подключающихся узлов.

В случае динамических ip-адресов, для возможности идентификации подключающихся узлов предусмотрен алгоритм и настройки программы приведенные ниже.

Подключающиеся узлы должны иметь 8-ми байтовый идентификатор. OPC104 запрашивает данный идентификатор посредством чтения двух данных с типом Bitstring32 (M_BO_NA_1). После получения 8 байтного идентификатора, OPC104 осуществляет поиск в таблице соответствия идентификаторов peer-ам. Если полученный идентификатор «связан» с каким либо peer-ом, то данный peer идентифицирован и дальнейший обмен осуществляется по обычной схеме. Общий адрес (COMMON ADDRESS) для идентификаторов всегда равен 1. Для поддержки данного алгоритма в программу введены следующие настройки.

- Собственный идентификатор задаваемый двумя 32 разрядными числами.

- Адреса первого и второго M_BO_NA_1 которые служат для передачи идентификатора.

- Каждый peer, с которым возможен обмен данными, имеет свой собственный идентификатор, задаваемый в настройках узла.

В случае когда предполагается, что ip-адрес будет получаться динамически, программа не должна делать какие-либо запросы (кроме STARTDT) до высылки своего идентификатора. Только ПОСЛЕ того как программа ответила на запросы адреса, программа может посылать запросы (например запрос General Interrogation). Данное ограничение возникает из-за того, что до того как узел идентифицирован, не имеет смысла отвечать на какие либо запросы, т.к. программа не знает кому она отвечает.

Собственный идентификатор узла и адреса первого и второго M_BO_NA_1 использующиеся для передачи идентификатора задаются в диалоге «Настройки программы».

Идентификатор peer-а задается в диалоге «Настройки узла»

Если в свойствах peer-а указано что режим подключения пассивный, и ID1 или ID2 не равны нулю, то программа только отображает ip-адрес узла в свойствах , но не сохраняет его в файле конфигурации iec104.xml.

Так как программа заранее не знает размеры полей **cause of transmission**, **common address**, **object address**, то эти размеры принимаются соответственно равными [1, 1, 2]. Стороннее устройство, подключающееся с динамическим ip-адресом должно иметь соответствующие длины полей. Для 2х программ OPC104, взаимодействующих друг с другом с использованием динамического ip-адреса, эти настройки так же следует указать равными [1, 1, 2].

Ниже приведен дамп процедуры идентификации со стороны узла воспринимающего входящее соединения от peer-а имеющего динамический ip-адрес в шестнадцатиричном виде с комментариями. При выполнении данного обмена заданы следующие настройки : адрес первого идентификатора – 7777, адрес второго идентификатора 7778. Идентификатор узла равен 22:44.

11-01-27 11:08:54 temp #RX: 68 04 07 00 00 00 // получили DATA_ACT

11-01-27 11:08:54 temp #TX: 68 04 0B 00 00 00 // ответили на DATA_ACT

11-01-27 11:08:54 temp #TX: 68 0A 00 00 00 00 66 01 05 01 61 1E // запрос на чтение 1:7777

11-01-27 11:08:54 temp #TX: 68 0A 02 00 00 00 66 01 05 01 62 1E // запрос на чтение 1:7778

11-01-27 11:08:54 temp #TX: 68 0B 04 00 00 00 64 01 06 01 00 00 14 // запрос GI

11-01-27 11:08:55 temp #RX: 68 0F 00 00 06 00 07 01 05 01 61 1E 16 00 00 00 00 // ответ 1:7777 (22)

11-01-27 11:08:55 temp #RX: 68 0F 02 00 06 00 07 01 05 01 62 1E 2C 00 00 00 00 // ответ 1:7778 (44)

Лицензирование программы OPC104

Программа OPC104 имеет два метода защиты от несанкционированного копирования. Программный метода и метод с использованием электронного ключа Guardant. Метод с применением электронного ключа предпочтителен, так как имеет несколько преимуществ по сравнению с программным методом защиты: он более надежен, он оставляет пользователю делать такие операции с дисками переустановка операционной системы и дефрагментация.

Программный метод защиты от несанкционированного копирования.

При первом запуске программы вы увидите диалог, показанный на рис. 11 .

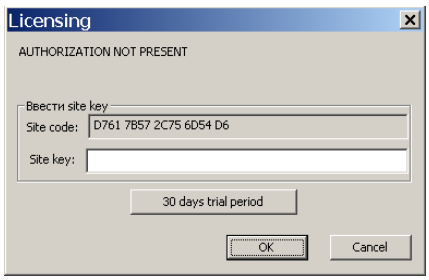


Рис. 11.

От Вас требуется ввести ключ, соответствующий указанному в строке ввода Site code. Получить этот ключ вы можете, запросив компанию Prosoft Systems по телефону или email и сообщив строку указанную в диалоге **лицензирование**. На рисунке эта строка равна D761 7857 2C75 6B54 B6. Введите полученный от компании **Prosoft Systems** код в строку ввода **Site key** и нажмите на кнопку **OK**. Если вы ввели правильный ключ, программа запустится. Следует помнить, что процедура лицензирования должна пройти в одной транзакции. Нельзя закрывать диалог лицензирования после того как вы выслали по email код так как при следующем запуске код может быть уже другим.

Процедура снятия программной лицензии.

Данная процедура может использоваться в случаях, когда необходимо перенести программу на другой компьютер или произвести такие операции как дефрагментирование или форматирование диска или переустановку операционной системы.

Для снятия программной лицензии выберите пункт меню **Лицензирование** -> Снятие лицензии и подтвердите Ваше намерение в появившемся диалоге.

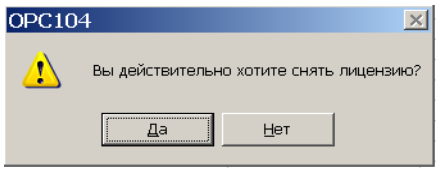
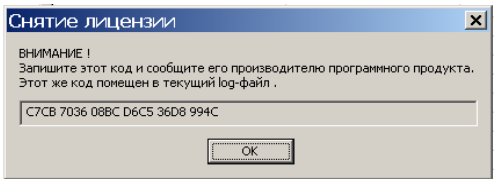


Рис. 12.

После подтверждения появится диалог в котором будет показан код снятия лицензии который следует сообщить компании ProsoftSystems с тем чтобы подтвердить то, что вы действительно сняли лицензию.



Программа дублирует полученный код снятия лицензии в текущем log файле. В этом файле появляется запись вида:

09-04-03 10:48:33 Код снятия лицензии: C7CB 7036 08BC D6C5 36D8 994C

После того как вы сняли лицензию вы можете выполнить запланированные операции (дефрагментация, форматирование, переустановка операционной системы) и затем снова запросить ключ лицензирования.

Защита от нелицензионного копирования с применением аппаратных USB-ключей Guardant.

Этот метод защиты требует наличия подключенного аппаратного ключа Guardant (может входить в комплект покупки). Предварительно следует установить драйвер Guardant. Данный драйвер входит в комплект поставки, но также его можно загрузить с сайта производителя www.guardant.ru.

Установка драйвера Guardant.

Пользователь, работающий с Windows 2008/Vista/2003/XP/2000, должен обладать правами администратора системы, иначе установка драйверов будет невозможна.

Чтобы установить драйверы Guardant, выполните следующие действия:

Отсоедините все ключи Guardant от портов компьютера.

Запустите файл INSTDRV.exe и следуйте указаниям мастера установки

После успешной установки драйверов подсоедините ключ Guardant к соответствующему порту.

Свидетельством того, что USB-ключ был успешно инициализирован операционной системой, является световая индикация ключа. Кроме того, ключ должен появиться в списке устройств Диспетчера оборудования Windows.