# **Стек протоколов DLMS:**

# **Формат сообщения**

## **HDLC кадр**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер в байтах | 1 | 2 | от 1 до 5 | 1 | 2 | до 2030 | 2 | 1 |
| HDLC кадр | FD | FF | AddressField | Control | HCS | Information | FCS | FD |

**FD** (**Frame Delimiter**) –флаг (**1 байт**), разделитель HDLC кадров. Все HDLC кадры должны начинаться и заканчиваться полем флага "**01111110**" (0x**7E**).

**FF** (**Frame Format**) – формат HDLC кадра (**2 байта**).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № бита | 15..12 | 11 | 10..8 | 7..0 |
| Frame Format | FrameType | Segmentation | RFU | Length |

**FrameType** – тип кадра, всегда "**1010**" (0xA). Возможные типы кадров представлены ниже:

|  |  |
| --- | --- |
| FrameType | Описание |
| A | (I frame) READ REQUEST |
| A | (I frame) READ REASPONSE |
| A | (RR frame) RECEIVE READY |

**Segmentation** – признак сегментирования кадра. Возможные значения поля представлены ниже:

|  |  |
| --- | --- |
| Segmentation | Описание |
| 0 | Единственный (последний) кадр |
| 1 | Есть следующий кадр |

**Length** – длина кадра в байтах. При расчете длины кадра флаги начала и конца кадра не учитываются.

**AddressField** – адресное поле (от 1 до 5 байт), определяет уникальные адреса получателя (DestinationAddress) и отправителя (SourceAddress), участвующих в передаче конкретного кадра.

При взаимодействии сервером является прибор учета (**Server**), клиентом выступает радиомодуль (**Client**).

Адрес клиента (**ClientAddress**) – это однобайтовое значение, он идентифицирует клиента.

Зарезервированные адреса клиента:

|  |  |
| --- | --- |
| ClientAddress | Описание |
| 0x00 | Нет станции. |
| 0x01 | Процесс управляется клиентом. |
| 0x10 | Общий доступ (самый низкий уровень безопасности). |

Адрес клиента используется для выбора уровня доступа.

Особенностью счетчиков Вектор-100 является наличие трех различных уровней доступа:

|  |  |  |
| --- | --- | --- |
| Уровень доступа | Адрес клиента | Права доступа |
| 0 | 0x10 | Открытый доступ, без пароля – читать все читаемые объекты. |
| 1 | 0x20  (определяется произаодителем) | Сбор данных со статическим паролем – читать все читаемые объекты, писать некоторые указанные для записи объекты (пример: установить дату, время, конец расчетного периода). |
| 1 | 0x01 | Управление доступом, со статическим паролем администрирования – чтение и запись всех объектов. |

При кодировании адреса клиента резервируется младший значащий бит, который затем устанавливается в единицу.

В нашем случае для адреса клиента используется значение 0x**20** (десятичное 32), после кодирования получаем 0x**41**:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № бита | MSB | 6 | 5 | 4 | 3 | 2 | 1 | LSB |
| ClientAddress | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

Адрес сервера (**ServerAddress**) – состоит из двух частей, верхняя часть (**upper** part) – это логический адрес устройства (**logical device address**), а нижняя часть (**lower** part) – это физический адрес устройства (**physical device address**).

Возможны три варианта адресации сервера:

* **Однобайтовая адресация**. Есть только верхний адрес. Это однобайтовое значение.

*В некоторых случаях, нижняя часть может быть опущена.* *Например, в режиме "точка-точка" в конфигурации с одним счетчиком, мы не заботимся о физическом адресе счетчика. Заметим, однако, что в нашем случае эта особенность не реализуется*.

* **Двухбайтовая адресация**. Есть верхний адрес – 1 байт и нижний адрес – 1 байт.
* **Четырехбайтовая адресация**. Есть верхний адрес – 2 байта и нижний адрес – два байта.

В нашем случае используется четырехбайтовая адресация.

|  |  |  |
| --- | --- | --- |
| Размер в байтах | 2 | 2 |
| ServerAddress | upper part | lower part |

При кодировании адреса сервера резервируется младший значащий бит каждого байта (бит расширения), которые затем устанавливаются в ноль, и только младший значащий бит нижней части адреса устанавливается в единицу.

Примечание:

*Адрес сервера может быть длиной 1, 2 или 4 байта. Для адресации серверов используется метод расширенной адресации, при этом адрес сервера может быть разделен на «Верхний» и «Нижний». «Верхний» адрес может быть адресом логического устройства внутри физического устройства, а «нижний» – адресом физического устройства при многоточечной конфигурации сети. «Верхний» адрес должен присутствовать обязательно, «Нижний» может отсутствовать. Признаком наличия «Нижнего» адреса является нулевой младший бит в байте «Верхнего» адреса. При однобайтовой адресации младший бит адреса должен быть установлен в «1», при многобайтовой адресации младшие биты всех байт, кроме последнего, должны быть установлены в «0», а у последнего – в «1». Содержимое адреса располагается в старших 7 битах каждого адреса, таким образом, адресное пространство при 1 байтовой адресации составляет от 0х00 до 0х7F, а при 2-х байтовой адресации – от 0х00 до 0х3FFF.*

Примечание:

*Верхняя и нижняя части адреса кодируются независимо друг от друга и уже в закодированном виде подставляются в поле адреса.*

В нашем случае верхняя часть адреса сервера 0x0001 (после кодирования 0x0002), нижняя часть адреса сервера 0x128C (после кодирования 0x4A19).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № бита | 7 | 6 | 5 | 4 | 3 | 2 | 1 | LSB | 7 | 6 | 5 | 4 | 3 | 2 | 1 | LSB |
| upper part | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № бита | 7 | 6 | 5 | 4 | 3 | 2 | 1 | LSB | 7 | 6 | 5 | 4 | 3 | 2 | 1 | LSB |
| lower part | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

**Control** – управляющее поле (1 байт), задает тип команды или ответа, а также значения счетчиков, отправленных/принятых кадров.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № бита | 7..5 | 4 | 3..1 | 0 |
| Control | RFNumber | PF | SFNumber | FrameType |

**RFNumber** – значение счетчика принятых кадров.

**PF** – бит бит опроса/завершения (poll/final). Устанавливается в единицу.

**SFNumber** – значение счетчика отправленных кадров.

**FrameType** – тип кадра. Возможные типы кадров и соответствующие им значения приведены ниже:

|  |  |  |
| --- | --- | --- |
| FrameType | Код | Описание |
| 0 | I | Информационный кадр (Information frame). |
| 1 | RR | Готов к приему (Receive ready) |
| 1 | RNR | Не готов к приему (Receive not ready) |
| 1 | SNRM | Установить режим нормального ответа (Set Normal Response Mode) |
| 1 | DISC | Разъединить (Disconnect) |
| 1 | UA | Ненумерованное подтверждение (Unnumbered Acknowledgment) |

**HCS** (**Header Check Sequence**) – Код целостности заголовка (2 байта).

**Information** – информационное поле. Максимальное значение длины информационного поля 2030 байт (значение по умолчанию 128 байт).

**FCS** (**Frame Check Sequence**) – Код целостности HDLC кадра (2 байта).

## **Информационное поле**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер в байтах | 3 | 1 | 1 | до 2025 |
| Information | LLC | APDU\_Type | APDU\_Length | APDU\_Block |

**LLC** (**Logical link control**) – байты логического управления каналом (3 байта). Возможные значения LLC приведены ниже:

|  |  |
| --- | --- |
| LLC | Описание |
| 0xE6E600 | Нисходящий канал: от клиента к серверу. |
| 0xE6E700 | Восходящий канал: от сервера к клиенту. |

**APDU\_Type** (**COSEM Application Layer Protocol Data Unit Type**) – тип блока данных COSEM протокола прикладного уровня (1 байт). Возможные типы блока данных COSEM и соответствующие им значения приведены ниже:

|  |  |
| --- | --- |
| APDU\_Type | Описание |
| 0x60 | COSEM AARQ-PDU: Кодирование метки для запроса прикладной ассоциации ([APPLICATION 0], Application) |
| 0x61 | COSEM AARE-PDU: Кодирование метки для подтверждения создания прикладной ассоциации ([APPLICATION 1], Application) |
| 0x05 | COSEM APDU: Запрос на чтение (Read Request) |
| 0x0C | COSEM APDU: Ответ на запрос чтения (Read Response) |

**APDU\_Length** – длины содержимого блок данных COSEM (1 байт).

*Смысл и характер длины содержимого зависит от типа блока данных COSEM, например, для COSEM AARQ-PDU указывается длина поля контента, для COSEM APDU указывается количество элементов в последовательности.*

**APDU\_Block** – блок данных COSEM (до 2025 байт)

# **Режимы открытия протокола**

Существует два возможных режима открытия протокола:

* МЭК 62056-21 Режим Е (IEC 62056-21 Mode E);
* Прямой HDLC (МЭК 62056-46/IEC 62056-46).

МЭК 62056-21 Режим E используется через оптический интерфейс, в то время как прямой HDLC реализуется только через электрический интерфейс. Прямой HDLC инициируется отправкой кадра «установить нормальный режим ответа» (Set Normal Response Mode – SNRM frame) (см. главу 4). В отличии от этого, МЭК 62056-21 режим Е должен сначала осуществить миграцию с МЭК 62056-21 в DLMS/COSEM (см. главу 3.1), и после этого осуществить отправку кадра SNMR.

## **Миграция с МЭК 62056-21 в DLMS/COSEM**

DLMS/COSEM предоставляет возможность миграции путем добавления нового режим протокола «E» к стандарту IEC 62056-21 (бывшему IEC 1107/МЭК 1107). При этом во время стартовой (открывающей) последовательности, счетчик (сервер) способен информировать (консультировать) клиента, что доступен дополнительный Режим E. Если клиент подтвердит (признает) его, они будут продолжать обмениваться данными с помощью 3-х уровневого, ориентированного на подключение, HDLC протокола. Обмен информацией осуществляется затем с использованием объектной модели COSEM. Если нет, обмен данными продолжается в обычном режиме, хотя функциональность может быть ограничена [§4.9, GB 5th].

Переход на протокол HDLC для прямого локального обмена данными, режим протокола Е показан в разделе 3.1.1 [§6, GB 5th].

### **Пример: IEC 62056-21 Mode E**

*Установить режим: 300 бод, 7 бит данных, контроль четности, 1 стоп-бит (7E1).*

|  |
| --- |
| **[Шаг #1] Client → Server: (REQUEST):** |
| 0x 2F 3F 30 30 37 35 33 37 34 38 21 0D 0A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2F | 3F | 30 30 37 35 33 37 34 38 | 21 | 0D | 0A |
| / | ? | Device Address (optional) = 00753748 | ! | <CR> | <LF> |
| (1) | (9) | (22) | (2) | (3) | (3) |

*Замечание: Адрес устройства – серийный номер счетчика.*

|  |
| --- |
| **[Шаг #1] Server → Client: (IDENTIFICATION):** |
| 0x 2F 45 47 4D 35 5C 32 47 33 42 31 34 34 54 38 4B 35 76 31 30 63 0D 0A |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2F | 45 47 4D | 35 | 5C | 32 | 47 33 42 31 34 34 54 38 4B 35 76 31 30 63 | 0D | 0A |
| / | EGM | 5 | \ | 2 | Identification = G3B144T8K5v10c | <CR> | <LF> |
| (1) | (12) | (13) | (23) | (24) | (14) | (3) | (3) |

*Замечание: строка идентификации содержит тип счетчика «G3B144» (всегда 6 символов) и версию прошивки «T8K5v10c» (всегда 8 символов).*

|  |
| --- |
| **[Шаг #2] Client → Server: (ACKNOWLEDGEMENT/OPTION SELECT):** |
| 0x 06 32 35 32 0D 0A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 06 | 32 | 35 | 32 | 0D | 0A |
| <ACK> | 2 | 5 | 2 | <CR> | <LF> |
| (4) | (10) | (13) | (11) | (3) | (3) |

*Изменить режим на 9600 бод, 7 бит данных, контроль четности, 1 стоп-бит (7E1).*

|  |
| --- |
| **[Шаг #2] Server → Client: (ACKNOWLEDGEMENT/SWITCH CONFIRM):** |
| 0x 2F 45 47 4D 35 5C 32 47 33 42 31 34 34 54 38 4B 35 76 31 30 63 0D 0A |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 06 | 32 | 35 | 32 | 0D | 0A |
| <ACK> | 2 | 5 | 2 | <CR> | <LF> |
| (4) | (10) | (13) | (11) | (3) | (3) |

*Изменить режим на 9600 бод, 8 бит данных, без контроля четности, 1 стоп-бит (8N1).*

Далее, клиент инициирует измерительный протокол HDLC, отправив кадр кадра «установить нормальный режим ответа» (SNRM frame).

### **Разъяснение содержания сообщения**

Разъяснение содержания сообщение:

|  |  |
| --- | --- |
| **№** | **Описание** |
| (1) | Символ начала «/» (косая черта (slash), код с 0x2f). |
| (2) | Символ конца «!» (восклицательный знак, код 0x21). |
| (3) | Символ завершения последовательности (CR, возврат каретки, код 0x0D; LF, перевод строки, код 0x0A). |
| (4) | Символ подтверждения (ACK, подтверждаете, код 0х06). |
| (9) | Команда «запрос передача» «?» (знак вопроса, код 0x3F). |
| (10) | Символ управления протоколом «V»:   * 0 – процедура нормального протокола; * 1 – процедура вторичного протокол; * 2 – процедура HDLC протокола; * 3...9 – зарезервированы для будущего использования. |
| (11) | Символ управления режимом «Y»:   * 0 – считывание данных; * 1 – режим программирования; * 2 – двоичный режим (HDLC); * 3...5 – зарезервированы для будущего использования; * 6...9 – используются по усмотрению производителя; * A…Z- зарезервированы для будущего использования. |
| (12) | Идентификация изготовителя, состоит из трех прописных букв. Эти буквы должны быть зарегистрированы администратором: флаг ассоциации (EGM для Elgama-Elektronika). Если устройство тарифицируется третья буква передается в нижнем регистре, минимальное время реакции устройства составляет 20 мс вместо 200 мс. Хотя если тарифное устройство передает прописной третий символ, это не означает поддержку времени реакции 20 мс. |
| (13) | Идентификация скорости передачи информации в бодах. Сообщения запроса, сообщения идентификации и сообщения подтверждения/выбора параметров передаются при начальной скорости 300 бод. Скорость передачи данных для сообщения данных зависит от скорости передачи информации в бодах определяемой протоколом:   * 0 – 300 бод; * 1 – 600 бод; * 2 – 1200 бод; * 3 – 2400 бод; * 4 – 4800 бод; * 5 – 9600 бод; * 6 – 19200 бод; * 7...9 – зарезервировано для последующего расширения. |
| (14) | Идентификация, определяется производителем, 16 печатных символов, за исключением «/» и «!». Обратный слеш «\» допускается только в качестве символа отмены (Escape). |
| (23) | Символ разделителя последовательности (Обратный слеш, код 0x5C), необязательное поле. Этот символ всегда следует за одно символьным полем (24). Это поле является частью 16-ти символьного поля идентификации (14). Несколько повторяющихся пар (23)/(24) не допускается. |
| (24) | Символ повышенной скорости передачи данных и идентификации режима «W» (необязательное поле). Это поле является частью 16 символьного поле идентификации (14). W должен быть зарегистрирован администратором: DLMS ассоциация пользователя (DLMS User Association). Использование символа «W» в режиме Е протокола:   * 0...1 – зарезервировано для будущего использования; * 2 – двоичный режим (HDLC); * 3...9 - зарезервированы для будущего использования. |

### **Упрощенный метод, только с одним изменением режима**

Так как строка «подтверждения/переключения подтверждения» (ACKNOWLEDGEMENT/SWITCH CONFIRM) является константой в программе сервера, он может быть легко переключен на скорость передачи данных в бодах и бинарный режим (Z baud, 8N1) одновременно. Символы «ACK 2 Z 2 CR LF» заменяются на их 8-битные аналоги, добавлением правильного (коректного) бита четности для того, чтобы имитировать эквивалент их «7E1». Этот альтернативный метод не виден клиенту; оба метода имеют эквивалентное поведение [§6.1, GB 5th].

# **Обмен кадрами SNRM/UA**

Установить режим нормального ответа (Set Normal Response Mode – SNRM) означает использование HDLC кадра для запроса установления MAC соединения. [§8.4.4.3.6, GB 5th] SNRM команда должна использоваться для перевода адресованной вторичной станции в нормальный режим ответа (NRM), где все поля управления должны быть длинной один октет. Вторичная станция должна подтвердить прием SNRM команды передачей ответа-подтверждения (Unnumbered acknowledge – UA) при первой возможности. После принятия этой команды, переменные состояния отправки и приема вторичной станции должны быть установлены в ноль (обнулены).

[§8.4.4.4.4.3.1, GB 5th] Главная станция должна инициализировать HDLC соединение со вторичной станцией, отправив SNRM команду и запустить функцию контроля времени отклика (response time-out function) (см. 8.4.4.4.10.1). Адресованная вторичная станция, получив SNRM команду правильно, направляет ответ UA при первой же возможности, и устанавливает в ноль свои переменные состояния отправки и получения. Если ответ UA получен правильно, HDLC соединение с адресованной вторичной станцией установлено, и первичная станция устанавливает в ноль свои соответствующие вторичной станции переменные состояния отправки и получения и останавливает функцию контроля времени отклика.

Если после получения SNRM команды вторичная станция определяет, что она не может включить (перейти в) указанный режим, то она должен отправить DM (Disconnect Mode - режим разъединения) ответ. Если ответ DM поступает правильно, первичная станция должна остановить функцию контроля времени отклика.

Если SNRM команда, UA ответ или DM ответ не получены правильно, то они должны быть проигнорированы. В результате этого функция контроля времени отклика на первичной станции будет работать, и первичная станция может повторно отправить SNRM команду и перезапустить функцию контроля времени отклика. Это действие может продолжаться до тех пор, пока не будет правильно получен UA или DM ответ или пока SNRM кадр не будет передан MAX\_NB\_OF\_RETRIES раз (см. 8.4.4.4.10.2). Все дальнейшие действия по восстановлению происходят на более высоком уровне.

Обмен SNRM/UA сообщениями позволяет не только создать соединение, но и дает возможность обсудить (согласовать) некоторые параметры согласования данных. Для целей настоящего протокола, два договорных параметра указанных в стандарте ИСО/МЭК 13239 (ISO/IEC 13239) были выбраны:

* Максимальная длина информационного поля.
* Размер окна.

Значение по умолчанию «максимальной длины информационного поля» составляет 128 байт. Максимальное значение зависит от качества физического канала (и может достигать 2030 байт).

Значение по умолчанию размер окна равно 1. Максимальное значение равно 7.

Согласование этих параметров осуществляется с помощью дополнительного информационного поля SNRM/UA кадров. Когда оно присутствует, то первые два байта идентификатор формата (Format Identifier 0x81) и идентификатор группы (Group Identifier 0х80) тоже всегда должны присутствовать. С другой стороны, любой (один или несколько) договорных параметров могут отсутствовать. В случае отсутствия определенного параметра следует использовать его значение по умолчанию. Отсутствие информационного поля подразумевает использование значений по умолчанию для каждого параметра.

## **Пример: SNRM/UA – Public client (lowest security level)**

Пример общего (открытого) клиента (самого низкого уровня безопасности).

|  |  |
| --- | --- |
| **Client → Server: (SNRM frame):** | |
| 0x 7E A0 0A 00 02 4A 19 21 93 D5 4E 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 0A | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x0A = 10 байт. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001 (Management Logical Device – управление логическим устройством). |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 21 | Адрес отправителя (клиент) = 0x10 =16 (Public client, lowest security level). |
| 93 | Тип кадра SNRM (Set Normal Response Mode) ([бит 0] = 1; бит опроса/завершения (PF): [бит 4] = 1 (poll bit [bit 4] = 1 (“true”))). |
|  | Информационное поле с HDLC параметрами отсутствует, предполагаются значения по умолчанию. |
| D5 4E | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |
| --- | --- |
| **Server → Client: (UA frame):** | |
| 0x 7E A0 15 21 00 02 4A 19 73 AF 75 81 80 06 05 01 83 06 01 4D 51 54 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 1F | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x15 = 21 байт. |
| 21 | Адрес получателя (клиент) = 0x10 =16 (Public client, lowest security level). |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 73 | Тип кадра UA (Unnumbered acknowledge), ([бит 0] = 1; бит опроса/завершения (PF): [бит 4] = 1 (final bit [bit 4] = 1 (“true”))). |
| AF 75 | Код целостности заголовка (HCS – Header Check Sequence). |
| 81 80 06 | Байты логического управления каналом (LLC bytes) (format identifier/group identifier/group length=0x06 = 6 octets). |
| 05 01 83 | Максимальная длина информационного поля для передачи (maximum information field length transmit = 0x83 = 131). |
| 06 01 4D | Максимальная длина информационного поля для приема (maximum information field length receive = 0x4D = 77). |
| 51 54 | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

## **Пример: SNRM/UA – low security level**

Пример низкого уровня безопасности.

|  |  |
| --- | --- |
| **Client → Server: (SNRM frame):** | |
| 0x 7E A0 0A 00 02 4A 19 41 93 80 2B 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 0A | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x0A = 10 байт. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001 (Management Logical Device – управление логическим устройством). |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32 (low security level). |
| 93 | Тип кадра SNRM [бит 0] = 1; счетчик отправленных кадров (SendNr): [биты 1..3] = 1; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 4. |
|  | Информационное поле с HDLC параметрами отсутствует, предполагаются значения по умолчанию. |
| 80 2B | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |
| --- | --- |
| **Server → Client: (UA frame):** | |
| 0x 7E A0 15 41 00 02 4A 19 73 1E F2 81 80 06 05 01 83 06 01 4D 51 54 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 1F | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x15 = 21 байт. |
| 21 | Адрес получателя (клиент) = 0x10 =16 (Public client, lowest security level). |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 73 | Тип кадра UA (Unnumbered acknowledge), ([бит 0] = 1; бит опроса/завершения (PF): [бит 4] = 1 (final bit [bit 4] = 1 (“true”))). |
| 1E F2 | Код целостности заголовка (HCS – Header Check Sequence). |
| 81 80 06 | Байты логического управления каналом (LLC bytes) (format identifier/group identifier/group length=0x06 = 6 octets). |
| 05 01 83 | Максимальная длина информационного поля для передачи (maximum information field length transmit = 0x83 = 131). |
| 06 01 4D | Максимальная длина информационного поля для приема (maximum information field length receive = 0x4D = 77). |
| 51 54 | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

# **Создание прикладной ассоциации**

## **Имена механизмов COSEM аутентификации**

[§ 9.3.3.7.2, GB 7th] Аутентификация (механизм проверки подлинности) является одним из аспектов безопасности предлагаемых спецификацией COSEM. В целях обеспечения различных уровней безопасности для поддержки механизма проверки подлинности, COSEM определяет три уровня безопасности:

* нет проверки подлинности (**самый низкий** уровень);
* **низкий** уровень, на основе проверки подлинности пароля (LLS – low level security), идентифицируется только клиент;
* **высокий** уровень, проверка подлинности на основе четырех паспортов (HLS – high level security), идентификация клиента и сервера. Счетчик Вектор-100 поддерживает «самый низкий» и «низкий» уровень безопасности («высокий уровень» не поддерживается).

## **Пример: AA using lowest security level**

В начале должен быть выполнен обмен SNRM/UA кадрами, для применения самого низкого уровня безопасности (lowest security level), (см. пример 4.1).

|  |  |  |  |
| --- | --- | --- | --- |
| **Client → Server: (I frame) Application Association Request (AARQ):** | | | |
| 0x 7E A0 2E 00 02 4A 19 21 10 B3 F4 E6 E6 00 60 1D A1 09 06 07 60 85 74 05 08 01 02 BE 10 04 0E 01 00 00 00 06 5F 1F 04 00 1C 03 20 00 00 38 5B 7E | | | |
| 7E | | Флаг открытия кадра HDLC. | |
| A0 2E | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x2E = 46 байт. | |
| 00 02 | | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 21 | | Адрес получателя (клиент) = 0x10 =16 (Public client, lowest security level). | |
| 10 | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 0; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 0. | |
| B3 F4 | | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E6 00 | | Байты логического управления каналом (LLC bytes). | |
|  | //---- BER (Basic Encoding Rules) кодирования AARQ APDU | | |
| 60 | | Кодирование тэга для AARQ-PDU ([APPLICATION 0], Application). | |
| 1D | | Кодирование поля длины содержимого AARQ (0x1D = 29 байт). | |
|  | //---- Кодирование компонента application-context-name (tagged component [1]) | | |
| A1 09 | | tag [1]: application-context-name; length =0x09 | |
|  | //---- Кодирование компонента application-context-name (OBJECT IDENTIFIER) | | |
| 06 | | Выбор для application-context-name (OBJECT IDENTIFIER, Universal) | |
| 07 | | Кодирование поля длины значения идентификатора объекта (7 октет) | |
| 60 85 74 05 08 01 02 | | | Идентификатор объекта (ссылка на серийный номер (SN referencing), шифрование не используется (no ciphering used)) |
|  | //---- Кодирование поля user-information (tagged component, [30]) | | |
| BE 10 | | tag [30] (Context-specific) user-information, length = 0x10 | |
|  | //---- Кодирование компонента application-context-name (OCTET STRING) | | |
| 04 | | Кодирование выбора для user-information (OCTET STRING) | |
| 0E | | Кодирование поля длины значения OCTET STRING (14 байт) | |
|  | //---- A-XDR кодирование DLMS Initiate.request-pdu | | |
| 01 | | Открытый тэг DLMS PDU CHOICE (InitiateRequest) | |
| 00 | | Флаг использования компонента dedicated-key (FALSE, not present) | |
| 00 | | Флаг использования компонента response-allowed (FALSE, default value conveyed) | |
| 00 | | Флаг использования компонента proposed-quality-of-service (FALSE, not present) | |
| 06 | | Компонент proposed-dlms-version-number (Unsigned8, value=6) | |
|  | //---- Кодирование блока соответствия (Conformance block) [APPLICATION 31] IMPLICIT BITSTRING (SIZE(24)) | | |
| 5F 1F | | Кодирование тэга [APPLICATION 31] (ASN.1 explicit tag) | |
| 04 | | Кодирование поля длина содержимого в октетах (4 байта) | |
| 00 | | Кодирование количества неиспользуемых битов в последнем (финальном) октете BITSTRING | |
| 1C 03 20 | | Предлагаемый блок соответствия (Conformance Block) | |
| 00 00 | | client-max-receive-pdu-size (unsigned16, value= not limited PDU size) | |
| 38 5B | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | | Флаг закрытия кадра HDLC. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Server → Client: (I frame) Application Association Response (AARE):** | | | |
| 0x 7E A0 3A 21 00 02 4A 19 30 47 7F E6 E7 00 61 29 A1 09 06 07 60 85 74 05 08 01 02 A2 03 02 01 00 A3 05 A1 03 02 01 00 BE 10 04 0E 08 00 06 5F 1F 04 00 18 02 20 00 50 FA 00 A1 55 7E | | | |
| 7E | | Флаг открытия кадра HDLC. | |
| A0 3A | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x3A = 54 байт. | |
| 21 | | Адрес получателя (клиент) = 0x10 =16. | |
| 00 02 | | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 30 | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 0; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 1. | |
| 47 7F | | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E7 00 | | Байты логического управления каналом (LLC bytes). | |
|  | //---- BER (Basic Encoding Rules) кодирования AARQ APDU | | |
| 61 | | Кодирование тэга для AARE-PDU ([APPLICATION 1], Application). | |
| 29 | | Кодирование поля длины содержимого AARE (41 октет). | |
|  | //---- Кодирование компонента application-context-name (tagged component [1]) | | |
| A1 | | Тэг для компонента application-context-name ([1], context-specific) | |
| 09 | | Длина поля значения целевого компонента. | |
| 06 | | Выбор для application-context-name (OBJECT IDENTIFIER, Universal) | |
| 07 | | Длина поля значения идентификатора объекта (7 октетов) | |
| 60 85 74 05 08 01 02 | | | Идентификатор объекта (SN referencing) |
|  | //---- Кодирование компонента result (tagged component [2]) | | |
| A2 | | Тэг для компонента result ([2], context-specific) | |
| 03 | | Длина поля значения целевого компонента. | |
| 02 | | Выбор для result (INTEGER, Universal) | |
| 01 | | Длина поля значения result (1 октет) | |
| 00 | | Значение result (0, accepted) | |
|  | //---- Кодирование компонента result-source-diagnostics (tagged component [3]) | | |
| A3 | | Тэг для компонента result-source-diagnostics | |
| 05 | | Длина поля значения целевого компонента. | |
| A1 | | Тэг для acse-service-user CHOICE(1) | |
| 03 | | Длина поля значения целевого компонента. | |
| 02 | | Выбор для result-source-diagnostics (INTEGER, Universal) | |
| 01 | | Длина поля значения result-source-diagnostics (1 октет) | |
| 00 | | Значение result-source-diagnostics (0, no diagnostics provided) | |
|  | //---- Кодирование поля компонента user-information (tagged component, [30]) | | |
| BE | | Тэг для поля компонента user-information ([30], Context-specific) | |
| 09 | | Длина поля значения целевого компонента. | |
| 04 | | Выбор для user-information (OCTET STRING, Universal) | |
| 0E | | Длина поля значения user-information (14 октетов) | |
|  | //---- A-XDR кодирование DLMS Initiate.response-pdu | | |
| 08 | | Открытый тэг DLMS PDU CHOICE (Initiate Response) | |
| 00 | | Флаг для proposed-quality-of-service component (FALSE, not present) | |
| 06 | | Компонент negotiated-dlms-version-number (Unsigned8, value=6) | |
| 5F 1F | | Кодирование тэга [APPLICATION 31] (ASN.1 explicit tag) | |
| 04 | | Длина поля содержимого (4 октета) | |
| 00 | | Количество неиспользуемых битов в последнем (финальном) октете битовой строки. | |
| 18 02 20 | | Договорной блок соответствия (Conformance Block) | |
| 00 50 | | server-max-receive-pdu-size (Unsigned16, value = 0x0050= 80 байт) | |
| FA 00 | | vaa-name: неизменное короткое имя интерфейсного класса Association SN | |
| A1 55 | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | | Флаг закрытия кадра HDLC. | |

## **Пример: AA с использованием low level authentication (LLS)**

В начале должен быть выполнен обмен SNRM/UA кадрами, для применения низкого уровня безопасности (low security level), (см. пример 4.2).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Client → Server: (I frame) Application Association Request (AARQ):** | | | | |
| 0x 7E A0 47 00 02 4A 19 41 10 5A 74 E6 E6 00 60 36 A1 09 06 07 60 85 74 05 08 01 02 8A 02 07 80 8B 07 60 85 74 05 08 02 01 AC 0A 80 08 4F 50 45 52 41 54 4F 52 BE 10 04 0E 01 00 00 00 06 5F 1F 04 00 1C 03 20 00 00 AD 96 7E | | | | |
| 7E | | Флаг открытия кадра HDLC. | | |
| A0 47 | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x47 = 71 байт. | | |
| 00 02 | | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. | | |
| 4A 19 | | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | | |
| 41 | | Адрес получателя (клиент) = 0x20 =32 (Data collection: with static password). | | |
| 10 | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 0; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 0. | | |
| 5A 74 | | Код целостности заголовка (HCS – Header Check Sequence). | | |
| E6 E6 00 | | Байты логического управления каналом (LLC bytes). | | |
|  | //---- BER (Basic Encoding Rules) кодирования AARQ APDU | | | |
| 60 | | Кодирование тэга для AARQ-PDU ([APPLICATION 0], Application). | | |
| 36 | | Кодирование поля длины содержимого AARQ (0x36 = 54 байта). | | |
|  | //---- Кодирование компонента application-context-name (tagged component [1]) | | | |
| A1 09 | | tag [1]: application-context-name; length =0x09 | | |
|  | //---- Кодирование компонента application-context-name (OBJECT IDENTIFIER) | | | |
| 06 | | Выбор для application-context-name (OBJECT IDENTIFIER, Universal) | | |
| 07 | | Кодирование поля длины значения идентификатора объекта (7 октет) | | |
| 60 85 74 05 08 01 02 | | | Идентификатор объекта (ссылка на серийный номер (SN referencing), шифрование не используется (no ciphering used)) | |
|  | //---- Кодирование поля компонента sender-acse-requirements (tagged component, [10]) | | | |
| 8A 02 | | tag [10] (IMPLICIT) ACSE-requirements ([Association Control Service Elements](http://yandex.ru/clck/jsredir?from=yandex.ru%3Bsearch%2F%3Bweb%3B%3B&text=&etext=1536.PkSMEVF5bavkpVf_TeygnB455IgCZ1oMhnkaWGqFKxUGeSbxdFRKwywJxICMIqi4.5684ba2ea1eede8c70d77325399ffb69b618cf5b&uuid=&state=PEtFfuTeVD5kpHnK9lio9dFa2ePbDzX7fH_cbK-eu2V8J4cbFpzDXWwD9FlNNMwOwSG-2u-pUKQ3XJJGh9ia_g,,&&cst=AiuY0DBWFJ5fN_r-AEszk0XcW5afl2FEWmgnLP3xen5YFmSnH4n5uTcZf4UmC1VWTZetfkhTg8XmUQTxpTbj0N0MGqfBVCGCU1cvG7f2wnzqAdPAovKZ5J4ItAiqtWKhi3lnAZgNNphhlkwGln9G7HW_2IWwNVcvF6DOHnbM3y765r3mXJ44vxLNwWybsnKkMKKbecpzL1R4eqn0qvVb-tGJvZKed7dTV2yRnMlKr6xLRhs-QO_lWWNlwAq9T4KSLExs311oq9U,&data=UlNrNmk5WktYejR0eWJFYk1Ldmtxal9CQVI1b0JWOGI2OEVPQkRpQzhkUVRyNXozUGFvS2ZzZnpScjdHbGg4UFQ4UzhoZmlOWEtUeDJ6VzVtdWNYeXByZWgxOG8zT29XdlV0azhtYW5yc3lGa0x6eXFMZDB0ZHFSa29lNzRsLUMtX05rQno2cUdITFlITlRfZVRIU1RnLCw,&sign=bc7dadc32adc0d6fe9ce57899dd69c9c&keyno=0&b64e=2&ref=orjY4mGPRjk5boDnW0uvlrrd71vZw9kpCyicf2ajz0hdnTeb_c6NKP_-UIyivUtCg1_VPlWbaCfjjgtzGDNMkJkVt5oEiaE7sEW1tbbeph3zC9J4NBpf6y86EhHdimRbR22UrLJZ91GPEbf7jvIoqt0ZvsMyuhzDkqMVTM-b1UY,&l10n=ru&cts=1504692201898&mc=4.914636847725354)); length= 0x02 | | |
| 07 | | Кодирование количества неиспользуемых битов в последнем байте BITSTRING | | |
| 80 | | Кодирование функционального блока проверки подлинности (0). | | |
|  | //---- Кодирование компонента mechanism-name (tagged component, [11]) | | | |
| 8B 07 | | tag [11] (IMPLICIT) mechanism-name, length= 0x07 | | |
| 60 85 74 05 08 02 01 | | | mechanism\_id(1): **low-level-security** | |
|  | //---- Кодирование компонента calling-authentication-value (tagged component, [12]) | | | |
| AC 0A | | tag [12]: calling-authentication-value, length= 0x0A | | |
| 80 | | Выбор для Authentication-value (charstring [0] IMPLICIT GraphicString) | | |
| 08 | | Длина поля Authentication-value (8 октетов). | | |
| 4F 50 45 52 41 54 4F 52 | | | | **Password (GraphicString “OPERATOR”)** |
|  | //---- Кодирование поля user-information (tagged component, [30]) | | | |
| BE 10 | | tag [30] (Context-specific) user-information, length = 0x10 | | |
|  | //---- Кодирование компонента application-context-name (OCTET STRING) | | | |
| 04 | | Кодирование выбора для user-information (OCTET STRING) | | |
| 0E | | Кодирование поля длины значения OCTET STRING (14 байт) | | |
|  | //---- A-XDR кодирование DLMS Initiate.request-pdu | | | |
| 01 | | Открытый тэг DLMS PDU CHOICE (InitiateRequest) | | |
| 00 | | Флаг использования компонента dedicated-key (FALSE, not present) | | |
| 00 | | Флаг использования компонента response-allowed (FALSE, default value conveyed) | | |
| 00 | | Флаг использования компонента proposed-quality-of-service (FALSE, not present) | | |
| 06 | | Компонент proposed-dlms-version-number (Unsigned8, value=6) | | |
|  | //---- Кодирование блока соответствия (Conformance block) [APPLICATION 31] IMPLICIT BITSTRING (SIZE(24)) | | | |
| 5F 1F | | Кодирование тэга [APPLICATION 31] (ASN.1 explicit tag) | | |
| 04 | | Кодирование поля длина содержимого в октетах (4 байта) | | |
| 00 | | Кодирование количества неиспользуемых битов в последнем (финальном) октете BITSTRING | | |
| 1C 03 20 | | Предлагаемый блок соответствия (Conformance Block) | | |
| 00 00 | | client-max-receive-pdu-size (unsigned16, value= not limited PDU size) | | |
| AD 96 | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | | |
| 7E | | Флаг закрытия кадра HDLC. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Server → Client: (I frame) Application Association Response (AARE):** | | | |
| 0x 7E A0 3A 41 00 02 4A 19 30 F6 F8 E6 E7 00 61 29 A1 09 06 07 60 85 74 05 08 01 02 A2 03 02 01 00 A3 05 A1 03 02 01 00 BE 10 04 0E 08 00 06 5F 1F 04 00 18 02 20 00 50 FA 00 A1 55 7E | | | |
| 7E | | Флаг открытия кадра HDLC. | |
| A0 3A | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x3A = 54 байт. | |
| 41 | | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 30 | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 0; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 1. | |
| F6 F8 | | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E7 00 | | Байты логического управления каналом (LLC bytes). | |
|  | //---- BER (Basic Encoding Rules) кодирования AARQ APDU | | |
| 61 | | Кодирование тэга для AARE-PDU ([APPLICATION 1], Application). | |
| 29 | | Кодирование поля длины содержимого AARE (41 октет). | |
|  | //---- Кодирование компонента application-context-name (tagged component [1]) | | |
| A1 | | Тэг для компонента application-context-name ([1], context-specific) | |
| 09 | | Длина поля значения целевого компонента. | |
| 06 | | Выбор для application-context-name (OBJECT IDENTIFIER, Universal) | |
| 07 | | Длина поля значения идентификатора объекта (7 октетов) | |
| 60 85 74 05 08 01 02 | | | Идентификатор объекта (SN referencing) |
|  | //---- Кодирование компонента result (tagged component [2]) | | |
| A2 | | Тэг для компонента result ([2], context-specific) | |
| 03 | | Длина поля значения целевого компонента. | |
| 02 | | Выбор для result (INTEGER, Universal) | |
| 01 | | Длина поля значения result (1 октет) | |
| 00 | | Значение result (0, accepted) | |
|  | //---- Кодирование компонента result-source-diagnostics (tagged component [3]) | | |
| A3 | | Тэг для компонента result-source-diagnostics | |
| 05 | | Длина поля значения целевого компонента. | |
| A1 | | Тэг для acse-service-user CHOICE(1) | |
| 03 | | Длина поля значения целевого компонента. | |
| 02 | | Выбор для result-source-diagnostics (INTEGER, Universal) | |
| 01 | | Длина поля значения result-source-diagnostics (1 октет) | |
| 00 | | Значение result-source-diagnostics (0, no diagnostics provided) | |
|  | //---- Кодирование поля компонента user-information (tagged component, [30]) | | |
| BE | | Тэг для поля компонента user-information ([30], Context-specific) | |
| 09 | | Длина поля значения целевого компонента. | |
| 04 | | Выбор для user-information (OCTET STRING, Universal) | |
| 0E | | Длина поля значения user-information (14 октетов) | |
|  | //---- A-XDR кодирование DLMS Initiate.response-pdu | | |
| 08 | | Открытый тэг DLMS PDU CHOICE (Initiate Response) | |
| 00 | | Флаг для proposed-quality-of-service component (FALSE, not present) | |
| 06 | | Компонент negotiated-dlms-version-number (Unsigned8, value=6) | |
| 5F 1F | | Кодирование тэга [APPLICATION 31] (ASN.1 explicit tag) | |
| 04 | | Длина поля содержимого (4 октета) | |
| 00 | | Количество неиспользуемых битов в последнем (финальном) октете битовой строки. | |
| 18 02 20 | | Договорной блок соответствия (Conformance Block) | |
| 00 50 | | server-max-receive-pdu-size (Unsigned16, value = 0x0050= 80 байт) | |
| FA 00 | | vaa-name: неизменное короткое имя интерфейсного класса Association SN | |
| A1 55 | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | | Флаг закрытия кадра HDLC. | |

## **Пример: AA с использованием LLS с пустым паролем (blank password)**

В данном примере прикладная ассоциация (АА) устанавливается, с использованием низкого уровня безопасности (LLS) с «пустым» паролем. Поскольку пароль еще не был установлен в приборе учета, счетчик принимает этот запрос прикладной ассоциации (Application Association Request).

В начале должен быть выполнен обмен SNRM/UA кадрами, для применения низкого уровня безопасности (low security level), (см. пример 4.2).

|  |  |  |  |
| --- | --- | --- | --- |
| **Client → Server: (I frame) Application Association Request (AARQ):** | | | |
| 0x 7E A0 42 00 02 52 DD 41 10 7B CB E6 E6 00 60 31 80 02 07 80 A1 09 06 07 60 85 74 05 08 01 02 8A 02 07 80 8B 07 60 85 74 05 08 02 01 AC 02 80 00 BE 0F 04 0D 01 00 00 00 06 5F 04 00 18 02 20 00 00 1C AE 7E | | | |
| 7E | | Флаг открытия кадра HDLC. | |
| A0 42 | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x42 = 66 байт. | |
| 00 02 | | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 52 DD | | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x14EE = 5358. | |
| 41 | | Адрес получателя (клиент) = 0x20 =32 (Data collection: with static password). | |
| 10 | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 0; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 0. | |
| 7B CB | | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E6 00 | | Байты логического управления каналом (LLC bytes). | |
|  | //---- BER (Basic Encoding Rules) кодирования AARQ APDU | | |
| 60 | | Кодирование тэга для AARQ-PDU ([APPLICATION 0], Application). | |
| 31 | | Кодирование поля длины содержимого AARQ (0x31 = 49 байта). | |
| 80 02 | | tag [0] (IMPLICIT BIT STRING) protocol-version; length= 0x02 | |
| 07 | | {version1 (0)} | |
|  | //---- Кодирование компонента application-context-name (tagged component [1]) | | |
| A1 09 | | tag [1]: application-context-name; length =0x09 | |
|  | //---- Кодирование компонента application-context-name (OBJECT IDENTIFIER) | | |
| 06 | | Выбор для application-context-name (OBJECT IDENTIFIER, Universal) | |
| 07 | | Кодирование поля длины значения идентификатора объекта (7 октет) | |
| 60 85 74 05 08 01 02 | | | Идентификатор объекта (ссылка на серийный номер (SN referencing), шифрование не используется (no ciphering used)) |
|  | //---- Кодирование поля компонента sender-acse-requirements (tagged component, [10]) | | |
| 8A 02 | | tag [10] (IMPLICIT) ACSE-requirements ([Association Control Service Elements](http://yandex.ru/clck/jsredir?from=yandex.ru%3Bsearch%2F%3Bweb%3B%3B&text=&etext=1536.PkSMEVF5bavkpVf_TeygnB455IgCZ1oMhnkaWGqFKxUGeSbxdFRKwywJxICMIqi4.5684ba2ea1eede8c70d77325399ffb69b618cf5b&uuid=&state=PEtFfuTeVD5kpHnK9lio9dFa2ePbDzX7fH_cbK-eu2V8J4cbFpzDXWwD9FlNNMwOwSG-2u-pUKQ3XJJGh9ia_g,,&&cst=AiuY0DBWFJ5fN_r-AEszk0XcW5afl2FEWmgnLP3xen5YFmSnH4n5uTcZf4UmC1VWTZetfkhTg8XmUQTxpTbj0N0MGqfBVCGCU1cvG7f2wnzqAdPAovKZ5J4ItAiqtWKhi3lnAZgNNphhlkwGln9G7HW_2IWwNVcvF6DOHnbM3y765r3mXJ44vxLNwWybsnKkMKKbecpzL1R4eqn0qvVb-tGJvZKed7dTV2yRnMlKr6xLRhs-QO_lWWNlwAq9T4KSLExs311oq9U,&data=UlNrNmk5WktYejR0eWJFYk1Ldmtxal9CQVI1b0JWOGI2OEVPQkRpQzhkUVRyNXozUGFvS2ZzZnpScjdHbGg4UFQ4UzhoZmlOWEtUeDJ6VzVtdWNYeXByZWgxOG8zT29XdlV0azhtYW5yc3lGa0x6eXFMZDB0ZHFSa29lNzRsLUMtX05rQno2cUdITFlITlRfZVRIU1RnLCw,&sign=bc7dadc32adc0d6fe9ce57899dd69c9c&keyno=0&b64e=2&ref=orjY4mGPRjk5boDnW0uvlrrd71vZw9kpCyicf2ajz0hdnTeb_c6NKP_-UIyivUtCg1_VPlWbaCfjjgtzGDNMkJkVt5oEiaE7sEW1tbbeph3zC9J4NBpf6y86EhHdimRbR22UrLJZ91GPEbf7jvIoqt0ZvsMyuhzDkqMVTM-b1UY,&l10n=ru&cts=1504692201898&mc=4.914636847725354)); length= 0x02 | |
| 07 | | Кодирование количества неиспользуемых битов в последнем байте BITSTRING | |
| 80 | | Кодирование функционального блока проверки подлинности (0). | |
|  | //---- Кодирование компонента mechanism-name (tagged component, [11]) | | |
| 8B 07 | | tag [11] (IMPLICIT) mechanism-name, length= 0x07 | |
| 60 85 74 05 08 02 01 | | | mechanism\_id(1): **low-level-security** |
|  | //---- Кодирование компонента calling-authentication-value (tagged component, [12]) | | |
| AC 02 | | tag [12]: calling-authentication-value, length= 0x02 | |
| 80 | | Выбор для Authentication-value (charstring [0] IMPLICIT GraphicString) | |
| 00 | | Длина поля Authentication-value (0 октетов) – **blank password**. | |
|  | //---- Кодирование поля user-information (tagged component, [30]) | | |
| BE 0F | | tag [30] (Context-specific) user-information, length = 0x0F = 15 | |
|  | //---- Кодирование компонента application-context-name (OCTET STRING) | | |
| 04 | | Кодирование выбора для user-information (OCTET STRING) | |
| 0D | | Кодирование поля длины значения OCTET STRING (13 байт) | |
|  | //---- A-XDR кодирование DLMS Initiate.request-pdu | | |
| 01 | | Открытый тэг DLMS PDU CHOICE (InitiateRequest) | |
| 00 | | Флаг использования компонента dedicated-key (FALSE, not present) | |
| 00 | | Флаг использования компонента response-allowed (FALSE, default value conveyed) | |
| 00 | | Флаг использования компонента proposed-quality-of-service (FALSE, not present) | |
| 06 | | Компонент proposed-dlms-version-number (Unsigned8, value=6) | |
|  | //---- Кодирование блока соответствия (Conformance block) [APPLICATION 31] IMPLICIT BITSTRING (SIZE(24)) | | |
| 5F | | Кодирование тэга [APPLICATION 31] (ASN.1 explicit tag) | |
| 04 | | Кодирование поля длина содержимого в октетах (4 байта) | |
| 00 | | Кодирование количества неиспользуемых битов в последнем (финальном) октете BITSTRING | |
| 18 02 20 | | Предлагаемый блок соответствия (Conformance Block) | |
| 00 00 | | client-max-receive-pdu-size (unsigned16, value= not limited PDU size) | |
| 1C AE | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | | Флаг закрытия кадра HDLC. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Server → Client: (I frame) Application Association Response (AARE):** | | | |
| 0x 7E A0 3A 41 00 02 52 DD 30 6B 16 E6 E7 00 61 29 A1 09 06 07 60 85 74 05 08 01 02 A2 03 02 01 00 A3 05 A1 03 02 01 00 BE 10 04 0E 08 00 06 5F 1F 04 00 18 02 20 00 50 FA 00 A1 55 7E | | | |
| 7E | | Флаг открытия кадра HDLC. | |
| A0 3A | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x3A = 54 байт. | |
| 41 | | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 52 DD | | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x14EE = 5358. | |
| 30 | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 0; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 1. | |
| 6B 16 | | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E7 00 | | Байты логического управления каналом (LLC bytes). | |
|  | //---- BER (Basic Encoding Rules) кодирования AARQ APDU | | |
| 61 | | Кодирование тэга для AARE-PDU ([APPLICATION 1], Application). | |
| 29 | | Кодирование поля длины содержимого AARE (41 октет). | |
|  | //---- Кодирование компонента application-context-name (tagged component [1]) | | |
| A1 | | Тэг для компонента application-context-name ([1], context-specific) | |
| 09 | | Длина поля значения целевого компонента. | |
| 06 | | Выбор для application-context-name (OBJECT IDENTIFIER, Universal) | |
| 07 | | Длина поля значения идентификатора объекта (7 октетов) | |
| 60 85 74 05 08 01 02 | | | Идентификатор объекта (SN referencing) |
|  | //---- Кодирование компонента result (tagged component [2]) | | |
| A2 | | Тэг для компонента result ([2], context-specific) | |
| 03 | | Длина поля значения целевого компонента. | |
| 02 | | Выбор для result (INTEGER, Universal) | |
| 01 | | Длина поля значения result (1 октет) | |
| 00 | | Значение result (**0, accepted**) | |
|  | //---- Кодирование компонента result-source-diagnostics (tagged component [3]) | | |
| A3 | | Тэг для компонента result-source-diagnostics | |
| 05 | | Длина поля значения целевого компонента. | |
| A1 | | Тэг для acse-service-user CHOICE(1) | |
| 03 | | Длина поля значения целевого компонента. | |
| 02 | | Выбор для result-source-diagnostics (INTEGER, Universal) | |
| 01 | | Длина поля значения result-source-diagnostics (1 октет) | |
| 00 | | Значение result-source-diagnostics (0, no diagnostics provided) | |
|  | //---- Кодирование поля компонента user-information (tagged component, [30]) | | |
| BE | | Тэг для поля компонента user-information ([30], Context-specific) | |
| 09 | | Длина поля значения целевого компонента. | |
| 04 | | Выбор для user-information (OCTET STRING, Universal) | |
| 0E | | Длина поля значения user-information (14 октетов) | |
|  | //---- A-XDR кодирование DLMS Initiate.response-pdu | | |
| 08 | | Открытый тэг DLMS PDU CHOICE (Initiate Response) | |
| 00 | | Флаг для proposed-quality-of-service component (FALSE, not present) | |
| 06 | | Компонент negotiated-dlms-version-number (Unsigned8, value=6) | |
| 5F 1F | | Кодирование тэга [APPLICATION 31] (ASN.1 explicit tag) | |
| 04 | | Длина поля содержимого (4 октета) | |
| 00 | | Количество неиспользуемых битов в последнем (финальном) октете битовой строки. | |
| 18 02 20 | | Договорной блок соответствия (Conformance Block) | |
| 00 50 | | server-max-receive-pdu-size (Unsigned16, value = 0x0050= 80 байт) | |
| FA 00 | | vaa-name: неизменное короткое имя интерфейсного класса Association SN | |
| A1 55 | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | | Флаг закрытия кадра HDLC. | |

В случае, если пароль уже задан в приборе учета, счетчик отклоняет этот запрос прикладной ассоциации (**AARQ**) отправкой соответствующего отклика (**AARE**) (т. е. некорректная попытка установления прикладной ассоциации):

|  |  |  |  |
| --- | --- | --- | --- |
| **Server → Client: (I frame) Application Association Response (AARE):** | | | |
| 0x 7E A0 3A 41 00 02 52 DD 30 6B 16 E6 E7 00 61 29 A1 09 06 07 60 85 74 05 08 01 02 A2 03 02 01 01 A3 05 A1 03 02 01 0D BE 10 04 0E 08 00 06 5F 1F 04 00 18 02 20 00 50 FA 00 B2 EC 7E | | | |
| 7E | | Флаг открытия кадра HDLC. | |
| A0 3A | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x3A = 54 байт. | |
| 41 | | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 52 DD | | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x14EE = 5358. | |
| 30 | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 0; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 1. | |
| 6B 16 | | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E7 00 | | Байты логического управления каналом (LLC bytes). | |
|  | //---- BER (Basic Encoding Rules) кодирования AARQ APDU | | |
| 61 | | Кодирование тэга для AARE-PDU ([APPLICATION 1], Application). | |
| 29 | | Кодирование поля длины содержимого AARE (41 октет). | |
|  | //---- Кодирование компонента application-context-name (tagged component [1]) | | |
| A1 | | Тэг для компонента application-context-name ([1], context-specific) | |
| 09 | | Длина поля значения целевого компонента. | |
| 06 | | Выбор для application-context-name (OBJECT IDENTIFIER, Universal) | |
| 07 | | Длина поля значения идентификатора объекта (7 октетов) | |
| 60 85 74 05 08 01 02 | | | Идентификатор объекта (SN referencing) |
|  | //---- Кодирование компонента result (tagged component [2]) | | |
| A2 | | Тэг для компонента result ([2], context-specific) | |
| 03 | | Длина поля значения целевого компонента. | |
| 02 | | Выбор для result (INTEGER, Universal) | |
| 01 | | Длина поля значения result (1 октет) | |
| 01 | | Значение result (**1, rejected-permanent**) | |
|  | //---- Кодирование компонента result-source-diagnostics (tagged component [3]) | | |
| A3 | | Тэг для компонента result-source-diagnostics | |
| 05 | | Длина поля значения целевого компонента. | |
| A1 | | Тэг для acse-service-user CHOICE(1) | |
| 03 | | Длина поля значения целевого компонента. | |
| 02 | | Выбор для result-source-diagnostics (INTEGER, Universal) | |
| 01 | | Длина поля значения result-source-diagnostics (1 октет) | |
| 0D | | Значение result-source-diagnostics (**0x0D = 13, authentication-failure**) | |
|  | //---- Кодирование поля компонента user-information (tagged component, [30]) | | |
| BE | | Тэг для поля компонента user-information ([30], Context-specific) | |
| 10 | | Длина поля значения целевого компонента. | |
| 04 | | Выбор для user-information (OCTET STRING, Universal) | |
| 0E | | Длина поля значения user-information (14 октетов) | |
|  | //---- A-XDR кодирование DLMS Initiate.response-pdu | | |
| 08 | | Открытый тэг DLMS PDU CHOICE (Initiate Response) | |
| 00 | | Флаг для proposed-quality-of-service component (FALSE, not present) | |
| 06 | | Компонент negotiated-dlms-version-number (Unsigned8, value=6) | |
| 5F 1F | | Кодирование тэга [APPLICATION 31] (ASN.1 explicit tag) | |
| 04 | | Длина поля содержимого (4 октета) | |
| 00 | | Количество неиспользуемых битов в последнем (финальном) октете битовой строки. | |
| 18 02 20 | | Договорной блок соответствия (Conformance Block) | |
| 00 50 | | server-max-receive-pdu-size (Unsigned16, value = 0x0050= 80 байт) | |
| FA 00 | | vaa-name: неизменное короткое имя интерфейсного класса Association SN | |
| B2 EC | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | | Флаг закрытия кадра HDLC. | |

## **Формат блока соответствия (Conformance block)**

[§ 9.4.5, GB 5th] Для того, чтобы включить оптимизированный COSEM сервер, потребуется реализация блока соответствия с добавленной расширенной функциональностью. COSEM блок соответствия можно отличить от стандартного блока соответствия по тегу «APPLICATION 31».

Счетчики Вектор-100 поддерживают блок соответствия = 0x18 02 20, т. е. READ (читать), WRITE (писать), MULTIPLE\_REFERENCES (множественные отношения), PARAMETERISED\_ACCESS (параметризированный доступ).

Таблица 5.1 Описание битов блока соответствия (Conformance block)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Бит** | **Описание** | | **Значение бита** | **Значение блока** |
| 23 | Зарезервировано (0) | Reserved (0) | 0 | 0x**1**80220 |
| 22 | Зарезервировано (0) | Reserved (0) | 0 |
| 21 | Зарезервировано (0) | Reserved (0) | 0 |
| 20 | Чтение | Read | 1 |
| 19 | Запись | Write | 1 | 0x1**8**0220 |
| 18 | Запись без подтверждения | Unconfirmed-write | 0 |
| 17 | Зарезервировано (0) | Reserved (0) | 0 |
| 16 | Зарезервировано (0) | Reserved (0) | 0 |
| 15 | Атрибут 0 с поддержкой установки (SET) | Attribute 0 supported with SET | 0 | 0x18**0**220 |
| 14 | Поддерживается приоритет управления | Priority management supported | 0 |
| 13 | Атрибут 0 с поддержкой запроса (GET) | Attribute 0 supported with GET | 0 |
| 12 | Передача блока с запросом | Block transfer with get | 0 |
| 11 | Передача блока с установкой | Block transfer with set | 0 | 0x180**2**20 |
| 10 | Передача блока с действием | Block transfer with action | 0 |
| 9 | Множественные отношения | Multiple references | 1 |
| 8 | Информационный отчет | Information report | 0 |
| 7 | Зарезервировано (0) | Reserved (0) | 0 | 0x1802**2**0 |
| 6 | Зарезервировано (0) | Reserved (0) | 0 |
| 5 | Параметризированный доступ | Parameterized access | 1 |
| 4 | Получить | Get | 0 |
| 3 | Установить | Set | 0 | 0x18022**0** |
| 2 | Избирательный доступ | Selective access | 0 |
| 1 | Уведомление о событии | Event notification | 0 |
| 0 | Действие | Action | 0 |

# **Передача данных**

## **Логическое имя устройства**

Логическое имя устройства (logical\_device\_name) используется для глобальной уникальной идентификации каждого логического устройства. Это октетная (байтовая) строка длиной до 16 октет. Первые три октета однозначно определяют производителя устройства (EGM for ElgamaElektronika). Они назначаются DLMS UA в сотрудничестве с FLAG Association. Производитель несет ответственность за обеспечение уникальности следующих октетов (до 13 октетов). Логическое имя устройства хранится в объекте COSEM logical device name – объекта данных с логическим именем 0-0:42.0.0\*255. При использовании ссылки на короткое имя, базовое имя объекта COSEM logical device name будет 0xFD00.

|  |  |  |
| --- | --- | --- |
| **OBIS** | **Base name** | **Interface Class (IC), Object description** |
| 0-0:42.0.0\*255 | 0xFD00 | (IC: Data) Logical device name |

### **COSEM Interface Class: Data (class\_id:1)**

[§ 4.2.1, BB 7th]. Объект «Data» хранит данные, относящиеся к внутренним (заводским) объектам прибора учета. Смысл значения (value) определяется логическим именем (logical\_name). Тип данных значения is CHOICE (выбор из массива). «Data» обычно используется для хранения конфигурации и параметров.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data (Class\_id=1)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. value | Static  Dynamic | Octet-string  CHOICE | X  X+0x08 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| Нет методов | |  |  |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Data» |
| value | Содержит текущие значения  CHOICE  {  *-- simple data types*  null-data [0],  boolean [3],  bit-string [4],  double-long [5],  double-long-unsigned [6],  octet-string [9],  visible-string [10],  utf8-string [12],  bcd (binary-coded decimal) [13],  integer [15],  long [16],  unsigned [17],  long-unsigned [18],  long64 [20],  long64-unsigned [21],  enum [22],  float32 [23],  float64 [24],  date-time [25],  date [26],  time [27].  *-- complex data types*  array [1],  structure [2],  compact-array [19].  }  Тип данных зависит от заложенного в logical\_name смысла (от семантики определяемого объекта), возможно, от производителя. Он должен быть выбран так, что бы вместе с logical\_name, однозначно интерпретировался. Любые простые и сложные типы данных могут быть использованы, если выбор по каким то причинам не будет ограничен. |

#### **Формат запроса**

|  |  |  |
| --- | --- | --- |
| Размер в байтах | 1 | 2 |
| APDU\_Block | VAS | ObjectShortName |

**VAS** (**Variable Access Specification**) – определение переменной доступа (1 байт). Возможные варианты определение переменной доступа приведены ниже:

|  |  |  |
| --- | --- | --- |
| VAS | CHOICE | Описание |
| 0x02 | CHOICE[2] | Переменная name |
|  |  |  |

**ObjectShortName** – короткое имя запрашиваемого объекта (2 байта).

|  |  |  |
| --- | --- | --- |
| Размер в байтах | 1 | 1 |
| ObjectShortName | BaseName | AttributeName |

**BaseName** – базовое имя интерфейсного класса (логическое имя объекта).

**AttributeName** – короткое имя атрибута.

|  |  |
| --- | --- |
| 0xFD08 | объект: Logical Device Name (0xFD00); атрибут: value (BaseName + 0x08) |

#### **Формат ответа**

|  |  |  |  |
| --- | --- | --- | --- |
| Размер в байтах | 1 | 2 | определяется форматом |
| APDU\_Block | SuccessFlag | DataFormat | Data |

**SuccessFlag** – признак успешного считывания (1 байт **0x00**).

**DataFormat** – определяет формат возвращаемых данных (2 байта).

|  |  |  |
| --- | --- | --- |
| Размер в байтах | 1 | 1 |
| DataFormat | DataType | DataLength |

**DataType** – тип данных (1 байт).

**DataLength** – длина данных (1 байт).

|  |  |
| --- | --- |
| 0x0A0B | Отображаемая строка размером 11 байт. |

Возможные типы данных приведены ниже:

|  |  |
| --- | --- |
| DataType | Описание |
| 0x0A | Отображаемая строка (visible string). |
|  |  |

Data – возвращаемые (запрошенные) данные.

|  |  |
| --- | --- |
| 0x45474D3735333734380000 | EGM753748 *(логическое имя устройства)* |

### **Пример считывания логического имени устройства**

В этом примере считывается логическое имя устройства (Logical Device Name).

|  |  |
| --- | --- |
| **Client → Server: (I frame) READ REQUEST:** | |
| 0x 7E A0 14 00 02 4A 19 41 32 EB AC E6 E6 00 05 01 02 FD 08 10 D3 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 14 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x14 = 20 байт. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| 32 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 1; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 1. |
| EB AC | Код целостности заголовка (HCS – Header Check Sequence). |
| E6 E6 00 | Байты логического управления каналом (LLC bytes). |
| 05 | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Запрос на чтение (Read Request). |
| 01 | Количество элементов в последовательности = 1. |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| FD 08 | Короткое имя (Short Name) = 0xFD00 (объект: Logical Device Name; атрибут: value) |
| 10 D3 | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |  |
| --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE:** | | |
| 0x 7E A0 1F 41 00 02 4A 19 52 C2 2B E6 E7 00 0C 01 00 0A 0B 45 47 4D 37 35 33 37 34 38 00 00 AD 9B 7E | | |
| 7E | Флаг открытия кадра HDLC. | |
| A0 1F | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x1F = 31 байт. | |
| 41 | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 52 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 1; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 2. | |
| C2 2B | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E7 00 | Байты логического управления каналом (LLC bytes). | |
| 0C | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Ответ на запрос чтения (Read Response). | |
| 01 | Количество элементов в последовательности = 1. | |
| 00 | Успешно (нет ошибок). | |
| 0A 0B | Отображаемая строка размером 11 байт (visible string (size(11))). | |
| 45 47 4D 37 35 33 37 34 38 00 00 | | EGM753748 |
| AD 9B | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | Флаг закрытия кадра HDLC. | |

**Примечание**: *Значение Logical Device Name «EGM753748» содержит идентификатор производителя «EGM» и серийный номер прибора учета «753748»*.

## **Объект ассоциации**

В DLMS/COSEM, объекты ассоциации играют роль «привратника» (сторожа), контролируют доступ к информации в приборе учета и предоставляют информацию если она доступна. Приборы учета Вектор-100 используют объект «ассоциация» для ссылки по короткому имени (Short Name (SN)), моделируемый интерфейсным классом «AssociationSN».

Логическое имя (OBIS идентификатро) объектов ассоциации [§ 4.6.1.1.23, BB 7th]:

|  |  |  |
| --- | --- | --- |
| **OBIS** | **Base name** | **Interface Class (IC), Object description** |
| 0-0:40.0.0\*255 | 0xFA00 | (IC: Association SN) Current association |

### **COSEM Interface Class: Association SN (class\_id:12)**

[§ 4.2.13, BB 7th]. Логические устройства COSEM способны создавать прикладные ассоциации в COSEM контексте, используя ссылку на короткое имя, моделирующую ассоциации через экземпляры класса «Association SN». Логические устройства COSEM могут иметь один экземпляр этого интерфейсного класса для каждой ассоциации, которую устройство способно поддерживать.

|  |  |  |  |
| --- | --- | --- | --- |
| **Association SN (Class\_id=12)** | | **Версия=1** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. object\_list | Static  Static | Octet-string  Objlist\_type | 0xFA00  0xFA08 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. reserved from previous versions 2. reserved from previous versions 3. read\_by\_logicalname (data) 4. get\_attributes&methods (data) 5. change\_LLS\_secret (data) 6. change\_HLS\_secret (data) 7. reserved from previous versions 8. reply\_to\_HLS\_authentication (data) | | Необязательный  Необязательный  Необязательный  Необязательный  Необязательный  Необязательный  Необязательный  Необязательный | -------  -------  0xFA30  0xFA38  0xFA40  0xFA48  -------  0xFA58 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

#### **Формат запроса**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размер в байтах | 1 | 2 | … | 1 | 2 |
| APDU\_Block | VAS | ObjectShortName |  | VAS | ObjectShortName |

**VAS** (**Variable Access Specification**) – определение переменной доступа (1 байт). Возможные варианты определение переменной доступа приведены ниже:

|  |  |  |
| --- | --- | --- |
| APDU\_Type | CHOICE | Описание |
| 0x02 | CHOICE[2] | Переменная name |
|  |  |  |

**ObjectShortName** – короткое имя запрашиваемого объекта (2 байта).

|  |  |  |
| --- | --- | --- |
| Размер в байтах | 1 | 1 |
| ObjectShortName | BaseName | AttributeName |

**BaseName** – базовое имя интерфейсного класса (логическое имя объекта).

**AttributeName** – короткое имя атрибута.

Количество блоков {VAS; ObjectShortName} определяется значением поля APDU\_Length (Количество элементов в последовательности).

|  |  |
| --- | --- |
| 0xFA00 | объект: Association SN (0xFA00); атрибут: logical\_name |
| 0xFA08 | объект: Association SN (0xFA00); атрибут: object\_list (BaseName + 0x08) |

#### **Формат ответа**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер в байтах | 1 | 2 | определяется форматом | … |
| APDU\_Block | SuccessFlag | DataFormat | Data |  |

**SuccessFlag** – признак успешного считывания (1 байт **0x00**).

**DataFormat** – определяет формат возвращаемых данных (2 байта).

|  |  |  |
| --- | --- | --- |
| Размер в байтах | 1 | 1 |
| DataFormat | DataType | DataLength |

**DataType** – тип данных (1 байт).

**DataLength** – длина данных (1 байт).

|  |  |
| --- | --- |
| 0x0A0B | Отображаемая строка размером 11 байт. |

Возможные типы данных приведены ниже:

|  |  |
| --- | --- |
| DataType | Описание |
| 0x09 | Байтовая строка (octet string). |
| 0x0182 | Массив объектов (objects array). |
| 0x10 | (integer 16) |
| 0x12 | (unsigned 16) |
| 0x11 | (unsigned 8) |

Data – возвращаемые (запрошенные) данные.

Количество блоков {SuccessFlag; DataFormat; Data} определяется значением поля APDU\_Length (Количество элементов в последовательности).

Второй блок имеет тип «**массив объектов**», объект представляет собой **структуру из четырех компонентов** и имеет следующую структуру:

* **base\_name** (2 байта).
* **class\_id** (2 байта).
* **version** (1 байт).
* **OBIS** (6 байт).

|  |  |  |
| --- | --- | --- |
| … | |  |
| 01 82 01 3D | | Массив из 0x013D = 317 объектов |
| 02 04 | Структура из четырех компонентов. | |
| 10 00 00 | (integer 16) base\_name = 0x0000. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 01 08 00 FF | | OBIS = 1-0:1.8.0\*255 (Суммарная энергия +A) |
| 02 04 | Структура из четырех компонентов. | |
| 10 01 00 | (integer 16) base\_name = 0x0100. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 01 08 01 FF | | OBIS = 1-0:1.8.1\*255 (Суммарная энергия +A по тарифу 1) |
| … | |  |

### **Пример считывания ассоциации (AssociationSN)**

В этом примере считывается список объектов объекта ассоциации.

|  |  |
| --- | --- |
| **Client → Server: (I frame) READ REQUEST:** | |
| 0x 7E A0 17 00 02 4A 19 41 54 B5 02 E6 E6 00 05 02 02 FA 00 02 FA 08 E8 F3 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 17 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x17 = 23 байта. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| 54 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 2; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 2. |
| B5 02 | Код целостности заголовка (HCS – Header Check Sequence). |
| E6 E6 00 | Байты логического управления каналом (LLC bytes). |
| 05 | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Запрос на чтение (Read Request). |
| 02 | Количество элементов в последовательности = 2. |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| FA 00 | Короткое имя (Short Name) = 0xFA00 (объект: Association SN; атрибут: logical\_name) |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| FA 08 | Короткое имя (Short Name) = 0xFA00 (объект: Association SN; атрибут: object\_list) |
| E8 F3 | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |  |
| --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE:** | | |
| 0x 7E A8 8D 41 00 02 4A 19 74 25 46 E6 E7 00 0C 02 00 09 06 00 00 28 00 00 FF 00 01 82 01 3D 02 04 10 00 00 12 00 03 11 00 09 06 01 00 01 08 00 FF 02 04 10 01 00 12 00 03 11 00 09 06 01 00 01 08 01 FF 02 04 10 02 00 12 00 03 11 00 09 06 01 00 01 08 02 FF 02 04 10 03 00 12 00 03 11 00 09 06 01 00 01 08 03 FF 02 04 10 04 00 12 00 03 11 00 09 06 01 00 01 08 04 FF 02 04 10 05 00 12 00 03 11 00 09 06 01 00 02 08 00 FF 02 04 93 9B 7E | | |
| 7E | Флаг открытия кадра HDLC. | |
| A8 8D | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 1, длина кадра = 0x8D=141. | |
| 41 | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 74 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 2; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 3. | |
| 25 46 | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E7 00 | Байты логического управления каналом (LLC bytes). | |
| 0C | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit): Ответ на запрос чтения (Read Response). | |
| 02 | Количество элементов в последовательности = 2. | |
| 00 | Успешно (нет ошибок). | |
| 09 06 | Строка байтов (октетов) размером 6 байт (octet string (size(6))). | |
| 00 00 28 00 00 FF | | OBIS = 0-0:40.0.0\*255 (AssociationSN) |
| 00 | Успешно (нет ошибок). | |
| 01 82 01 3D | | Массив из 0x013D = 317 объектов |
| 02 04 | Структура из четырех компонентов. | |
| 10 00 00 | (integer 16) base\_name = 0x0000. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 01 08 00 FF | | OBIS = 1-0:1.8.0\*255 (Суммарная энергия +A) |
| 02 04 | Структура из четырех компонентов. | |
| 10 01 00 | (integer 16) base\_name = 0x0100. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 01 08 01 FF | | OBIS = 1-0:1.8.1\*255 (Суммарная энергия +A по тарифу 1) |
| 02 04 | Структура из четырех компонентов. | |
| 10 02 00 | (integer 16) base\_name = 0x0200. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 01 08 02 FF | | OBIS = 1-0:1.8.2\*255 (Суммарная энергия +A по тарифу 2) |
| 02 04 | Структура из четырех компонентов. | |
| 10 03 00 | (integer 16) base\_name = 0x0300. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 01 08 03 FF | | OBIS = 1-0:1.8.2\*255 (Суммарная энергия +A по тарифу 3) |
| 02 04 | Структура из четырех компонентов. | |
| 10 04 00 | (integer 16) base\_name = 0x0400. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 01 08 04 FF | | OBIS = 1-0:1.8.4\*255 (Суммарная энергия +A по тарифу 4) |
| 02 04 | Структура из четырех компонентов. | |
| 10 05 00 | (integer 16) base\_name = 0x0500. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 02 08 00 FF | | OBIS = 1-0:2.8.0\*255 (Суммарная энергия -A) |
| 02 04 | Структура из четырех компонентов. | |
| 93 9B | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | Флаг закрытия кадра HDLC. | |

|  |  |
| --- | --- |
| **Client → Server: (RR frame) RECEIVE READY:** | |
| 0x 7E A0 0A 00 02 4A 19 41 71 9C EF 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 0A | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x0A=10 байт. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| 71 | Тип кадра RR [бит 0..3] = 1; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 3. |
| 9C EF | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |  |
| --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE:** | | |
| 0x 7E A8 8A 41 00 02 4A 19 76 2F A2 10 06 00 12 00 03 11 00 09 06 01 00 02 08 01 FF 02 04 10 07 00 12 00 03 11 00 09 06 01 00 02 08 02 FF 02 04 10 08 00 12 00 03 11 00 09 06 01 00 02 08 03 FF 02 04 10 09 00 12 00 03 11 00 09 06 01 00 02 08 04 FF 02 04 10 0A 00 12 00 03 11 00 09 06 01 00 05 08 00 FF 02 04 10 0B 00 12 00 03 11 00 09 06 01 00 05 08 01 FF 02 04 10 0C 00 12 00 03 11 00 09 06 01 00 05 08 02 FF 02 04 93 D3 7E | | |
| 7E | Флаг открытия кадра HDLC. | |
| A8 8A | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 1, длина кадра = 0x8A=138. | |
| 41 | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 76 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 3; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 3. | |
| 2F A2 | Код целостности заголовка (HCS – Header Check Sequence). | |
| 10 06 00 | (integer 16) base\_name = 0x0600. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 02 08 01 FF | | OBIS = 1-0:2.8.1\*255 (Суммарная энергия -A по тарифу 1) |
| 02 04 | Структура из четырех компонентов. | |
| 10 07 00 | (integer 16) base\_name = 0x0700. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 02 08 02 FF | | OBIS = 1-0:2.8.2\*255 (Суммарная энергия -A по тарифу 2) |
| 02 04 | Структура из четырех компонентов. | |
| 10 08 00 | (integer 16) base\_name = 0x0800. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 02 08 03 FF | | OBIS = 1-0:2.8.3\*255 (Суммарная энергия -A по тарифу 3) |
| 02 04 | Структура из четырех компонентов. | |
| 10 09 00 | (integer 16) base\_name = 0x0900. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 02 08 04 FF | | OBIS = 1-0:2.8.4\*255 (Суммарная энергия -A по тарифу 4) |
| 02 04 | Структура из четырех компонентов. | |
| 10 0A 00 | (integer 16) base\_name = 0x0A00. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 05 08 00 FF | | OBIS = 1-0:5.8.0\*255 (Суммарная энергия R1) |
| 02 04 | Структура из четырех компонентов. | |
| 10 0B 00 | (integer 16) base\_name = 0x0B00. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 05 08 01 FF | | OBIS = 1-0:5.8.1\*255 (Суммарная энергия R1 по тарифу 1) |
| 02 04 | Структура из четырех компонентов. | |
| 10 0C 00 | (integer 16) base\_name = 0x0C00. | |
| 12 00 03 | (unsigned 16) class\_id=3 (Register). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 05 08 02 FF | | OBIS = 1-0:5.8.2\*255 (Суммарная энергия R1 по тарифу 2) |
| 02 04 | Структура из четырех компонентов. | |
| 93 D3 | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | Флаг закрытия кадра HDLC. | |

|  |  |
| --- | --- |
| **Client → Server: (RR frame) RECEIVE READY:** | |
| 0x 7E A0 0A 00 02 4A 19 41 91 92 08 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 0A | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x0A=10 байт. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| 91 | Тип кадра RR [бит 0..3] = 1; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 4. |
| 92 08 | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

Здесь несколько «READ REASPONSE» и «RECEIVE READY» кадров данных будут опущены, и только последний «READ REASPONSE» кадр с сервера представлен ниже:

|  |  |  |
| --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE:** | | |
| 0x 7E A0 40 41 00 02 4A 19 7E 97 AB 10 E3 80 12 00 01 11 00 09 06 01 00 60 84 03 08 02 04 10 E3 90 12 00 01 11 00 09 06 01 00 60 84 03 09 02 04 10 DF 00 12 00 01 11 00 09 06 01 00 60 85 00 FF D8 2A 7E | | |
| 7E | Флаг открытия кадра HDLC. | |
| A0 40 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x40=64. | |
| 41 | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 7E | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 7; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 3. | |
| 97 AB | Код целостности заголовка (HCS – Header Check Sequence). | |
| 10 E3 80 | (integer 16) base\_name = 0xE380. | |
| 12 00 01 | (unsigned 16) class\_id=1 (Data). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 60 84 03 08 | | OBIS = 1-0:96.132.3\*8 ( ) |
| 02 04 | Структура из четырех компонентов. | |
| 10 E3 90 | (integer 16) base\_name = 0xE390. | |
| 12 00 01 | (unsigned 16) class\_id=1 (Data). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 02 08 02 FF | | OBIS = 1-0:96.132.3\*9 ( ) |
| 02 04 | Структура из четырех компонентов. | |
| 10 DF 00 | (integer 16) base\_name = 0xDF00. | |
| 12 00 01 | (unsigned 16) class\_id=1 (Data). | |
| 11 00 | (unsigned 8) version = 0. | |
| 09 06 | octet string [6]. | |
| 01 00 60 85 00 FF | | OBIS = 1-0:96.133.0\*255 ( ) |
| D8 2A | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | Флаг закрытия кадра HDLC. | |

## **Профиль загрузки**

Профиль загрузки моделируется интерфейсным классом «Profile generic».

|  |  |  |
| --- | --- | --- |
| **OBIS** | **Base name** | **Interface Class (IC), Object description** |
| 1-0:99.1.0\*255 | 0xC000 | (IC: Profile Generic) Load profile |

**Замечание**: *Логическое имя (код OBIS) однозначно определяет (идентифицирует) смысл информации, представленной объектом, независимым от производителя образом. Фактические значения коротких имен (base names) зависят от количества и типа объектов, созданных в устройстве и стратегии ассоциирования использованной производителем, поэтому они могут быть разными в различных устройствах.*

### **COSEM Interface Class: Profile generic (class\_id:7)**

[§ 4.2.6, BB 7th]. Класс «Profile generic» (общий профиль) определяет обобщенную концепцию хранения значений, полученных в ходе динамического опроса объектов. Объектами опроса (захвата) являются соответствующие атрибуты или элементы атрибутов COSEM объектов.

|  |  |  |  |
| --- | --- | --- | --- |
| **Profile generic (Class\_id=7)** | | **Версия=1** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. buffer 3. capture\_objects 4. capture\_period 5. sort\_method 6. sort\_object 7. entries\_in\_use 8. profile\_entries | Static  Dynamic  Static  Static  Static  Static  Dynamic  Static | Octet-string  Compact array or array  Array  Double-long-unsigned  Enumerated  Object\_definition  Double-long-unsigned  Double-long-unsigned | X  X+0x08  X+0x10  X+0x18  X+0x20  X+0x28  X+0x30  X+0x38 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. reset (data) 2. capture (data) 3. reserved from previous versions 4. reserved from previous versions | | Необязательный  Необязательный  Необязательный  Необязательный | X+0x58  X+0x60  -------  ------- |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибуты** | | | |
| logical\_name | | Идентифицирует экземпляры объектов IC «Profile generic» | |
| buffer | | Содержит последовательность записей. Каждая запись содержит значения захваченных объектов.  compact-array или array entry  entry::= structure  {  CHOICE  {  *-- simple data types*  null-data [0],  boolean [3],  bit-string [4],  double-long [5],  double-long-unsigned [6],  octet-string [9],  visible-string [10],  utf8-string [12],  bcd (binary-coded decimal) [13],  integer [15],  long [16],  unsigned [17],  long-unsigned [18],  long64 [20],  long64-unsigned [21],  enum [22],  float32 [23],  float64 [24],  date-time [25],  date [26],  time [27].  *-- complex data types*  array [1],  structure [2],  compact-array [19].  }  }  Количество и порядок элементов структуры поддерживает записи в том виде, как задано в определении capture\_objects. Буфер заполняется автоматическими захватами или последовательными вызовами метода «захватить» (capture (data)). Последовательность записей в массиве упорядочивается согласно указанному в sort\_method порядку.  По умолчанию: буфер (buffer) пуст после сброса.  Замечание 1: *Чтение всего буфера возвращает только записи с признаком «используются» (in use).*  Замечание 2: *Значение захваченного объекта может быть заменено на нулевые данные (null-data), если он может быть однозначно восстановлен путем сравнения с предыдущим значением (например для времени: если оно может быть рассчитана на основании предыдущего значения и периода захвата (capture\_period); или для значения: если оно равно предыдущему значению).*  Может быть доступен выборочный доступ (см. 4.4) к атрибутам буфера (опционально). Выборочные параметры доступа определены ниже. | |
| capture\_objects | | Определяет список объектов захвата, назначенных экземпляру объекта.  При вызове метода захвата данных (capture (data)) или автоматически в определенные интервалы, выбранные атрибуты копируются в буфер профиля.  Массив (array) capture\_object\_definition  capture\_object\_definition::= structure  {  class\_id: long-unsigned,  logical\_name: octet-string,  attribute\_index: integer,  data\_index: long-unsigned  }  - где attribute\_index является указателем на атрибут внутри объекта, идентифицируется по class\_id и logical\_name. attribute\_index 1 ссылается на 1-ый атрибут (т.е. logical\_name), attribute\_index 2 на 2-ой и т. д.; attribute\_index 0 ссылается на все общедоступные атрибуты;  - где data\_index – указатель, выбирающий определенный элемент атрибута. Первый элемент в структуре атрибутов определяется data\_index 1. Если атрибут не является структурой, то data\_index не имеет смысла. Если объект захвата является буфером профиля, то data\_index определяет захваченный из буфера объект (т.е. столбец) внутреннего профиля;  data\_index 0 ссылается на весь атрибут. | |
| capture\_period | | > = 1: предполагается автоматический захват. Задает период захвата в секундах.  0: без автоматического захвата; захват инициируется извне или события захвата происходят асинхронно. | |
| sort\_method | | Если профиль несортированный, он работает как fifo буфер («первым пришел-первым ушел», это значит, буфер сортируется по факту захвата, и не обязательно по времени хранящемся в объекте Clock, «часы»). Если буфер заполнен, следующий вызов capture() выталкивает первую (самую старую) запись буфера, чтобы освободить место для новой записи.  Если профиль отсортирован, вызов capture() сохраняет новую запись в соответствующую позицию в буфере, перемещая все следующие записи и, вероятно, теряя наименее интересную запись. Если новая запись войдет в буфер после последней записи и если буфер будет полностью заполнен, новая запись не будет сохранена вообще.  enum:  (1) fifo (first in first out, первым пришел – первым ушел),  (2) lifo (last in first out, последним пришел – первым ушел) ,  (3) largest (по увеличению),  (4) smallest (по уменьшению),  (5) nearest\_to\_zero (по близости к нулю),  (6) farest\_from\_zero (по удалению от нуля).  По умолчанию: fifo | |
| sort\_object | | Если профиль отсортирован, этот атрибут указывает что, регистр или часы, является основой для сортировки.  capture\_object\_definition см. выше.  По умолчанию: нет объекта для сортировки (только с sort\_method fifo или lifo).  Замечание 1: *Если sort\_method является FIFO или LIFO, то все элементы capture\_object\_definition, определяющие объект сортировки, могут быть равны нулю*. | |
| entries\_in\_use | | Подсчитывает количество записей, хранящихся в буфере. После вызова метод reset(), буфер не содержит каких-либо записей, и это значение будет нулевым. При каждом последующем вызове метода capture() это значение увеличивается до максимального числа записей, которые будут сохраняться (см. profile\_entries).  double-long-unsigned 0...profile\_entries  По умолчанию: 0. | |
| profile\_entries | | Указывает, сколько записей должно храниться в буфере.  double-long-unsigned 1 ... (ограничивается физическим размером).  По умолчанию: 1. | |
| **Параметры выборочного доступа (selective access) к атрибуту buffer** | | | |
| Access selector | Access parameter | | Comment |
| 1 | range\_descriptor | | только элементы буфера, соответствующие range\_descriptor должны быть возвращены в ответе. |
| 2 | entry\_descriptor | | только элементы буфера, соответствующие entry\_descriptor должны быть возвращены в ответе. |
|  | | range\_descriptor::= structure  {  restricting\_object: capture\_object\_definition  Определяет capture\_object ограничивающий диапазон извлекаемых записей. Разрешены только простые типы данных.  from\_value: возвращаются старейшие или наименьшие записи  CHOICE  { -- simple data types  double-long [5],  double-long-unsigned [6],  octet-string [9],  visible-string [10],  utf8-string [12],  integer [15],  long [16],  unsigned [17],  long-unsigned [18],  long64 [20],  long64-unsigned [21],  float32 [23],  float64 [24],  date-time [25],  date [26],  time [27].  }  to\_value: возвращаются новейшие или наибольшие записи  CHOICE {see above}  selected\_values: array capture\_object\_definition  Список столбцов для извлечения. Если массив пуст (не содержит записей), возвращаются все записанные данные. В противном случае, возвращаются только столбцы, указанные в массиве.  Тип capture\_object\_definition указан выше (capture\_objects).  }  entry\_descriptor::= structure  {  from\_entry: double-long-unsigned  Возвращается первая запись,  to\_entry: double-long-unsigned  Возвращается последняя запись,  to\_entry == 0: максимально доступная запись,  from\_selected\_value: long-unsigned  Индекс первого значения для извлечения,  to\_selected\_value: long-unsigned  Индекс последнего значения для извлечения,  to\_selected\_value == 0: максимально доступное selected\_value  }  Замечание 2: *from\_entry и to\_entry идентифицируют строки, from\_selected\_value to\_selected\_value идентифицируют столбцы буфера для извлечения.*  Замечание 3: *нумерация записей и выбранных значений начинается с 1*. | |
| **Методы** | | | |
| reset (data) | | Этот метод очищает буфер. После этого он не имеет допустимых записей;  После вызова этого метода entries\_in\_use равно 0. Этот вызов не вызывает никаких дополнительных операций с захваченными объектами. В частности, он не сбрасывает ни какие захваченные атрибуты.  data:: = integer(0) | |
| capture (data) | | Копирует значения объектов для захвата в буфер, считывая каждый объект захвата. В зависимости от sort\_method и фактического состояния буфера будет создана новая запись или заменена менее значимая запись. Пока не все записи еще использованы (обработаны), атрибут entries\_in\_use будет увеличиваться.  Вызов этого метода не вызывает каких-либо дополнительных операций с захваченными объектами, таких как capture()или reset().  Обратите внимание, если необходимо захватить несколько атрибутов объекта, они должны быть определены по одному в списке capture\_objects. Если attribute\_index = 0, все атрибуты будут захвачены. | |
| reserved from previous versions | | Зарезервировано для совместимости с предыдущими версиями. | |
| reserved from previous versions | | Зарезервировано для совместимости с предыдущими версиями. | |
|  | |  | |
| **Поведение объекта после модификации определенных атрибутов** | | | |
| Любое изменение одного из capture\_objects, описывающих статическую структуру буфера будет автоматически вызывать reset() и этот вызов распространяется на все другие профили захвата данного профиля.  Попытка записи в profile\_entries слишком большого для буфера значения, приведет к отклонению этой записи. | | | |
| **Ограничения** | | При определении объектов захвата следует избегать круговой ссылки на профиль. | |
| **Профиль, используемый для определения подмножества предпочтительных значений считывания** | | | |
| Установив profile\_entries в 1, объект Profile generic может использоваться, для определения набора предпочтительных значения считывания. См. также 6.2.17. Установка значения capture\_period равным 1 гарантирует, что значения обновляются каждую секунду. | | | |

#### **Формат запроса**

|  |  |  |  |
| --- | --- | --- | --- |
| Размер в байтах | 1 | 2 | … |
| APDU\_Block | VAS | ObjectShortName |  |

**VAS** (**Variable Access Specification**) – определение переменной доступа (1 байт). Возможные варианты определение переменной доступа приведены ниже:

|  |  |  |
| --- | --- | --- |
| APDU\_Type | CHOICE | Описание |
| 0x02 | CHOICE[2] | Переменная name |
|  |  |  |

**ObjectShortName** – короткое имя запрашиваемого объекта (2 байта).

|  |  |  |
| --- | --- | --- |
| Размер в байтах | 1 | 1 |
| ObjectShortName | BaseName | AttributeName |

**BaseName** – базовое имя интерфейсного класса (логическое имя объекта).

**AttributeName** – короткое имя атрибута.

Количество блоков {VAS; ObjectShortName} определяется значением поля APDU\_Length (Количество элементов в последовательности).

|  |  |
| --- | --- |
| 0xC010 | объект: Load Profile; атрибут: capture\_objects |
| 0xC018 | объект: Load Profile; атрибут: capture\_period |

#### **Формат ответа**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер в байтах | 1 | 2 | определяется форматом | … |
| APDU\_Block | SuccessFlag | DataFormat | Data |  |

**SuccessFlag** – признак успешного считывания (1 байт **0x00**).

**DataFormat** – определяет формат возвращаемых данных (2 байта).

|  |  |  |
| --- | --- | --- |
| Размер в байтах | 1 | 1 |
| DataFormat | DataType | DataLength |

**DataType** – тип данных (1 байт).

**DataLength** – длина данных (1 байт).

|  |  |
| --- | --- |
| 0x0A0B | Отображаемая строка размером 11 байт. |

Возможные типы данных приведены ниже:

|  |  |
| --- | --- |
| DataType | Описание |
| 0x01 | Массив объектов (objects array). |
| 0x02 | Структура (structure). |
| 0x12 | (unsigned 16) |
| 0x0F | (integer 8). |
| 0x06 | (unsigned 32) |
| 0x09 | Байтовая строка (octet string). |

Data – возвращаемые (запрошенные) данные.

Количество блоков {SuccessFlag; DataFormat; Data} определяется значением поля APDU\_Length (Количество элементов в последовательности).

Второй блок имеет тип «**массив объектов**», объект представляет собой **структуру из четырех компонентов** и имеет следующую структуру:

* **class\_id** (2 байта).
* **OBIS** (6 байт).
* **Attribute index** (1 байт).
* **Data index** (2 байта).

|  |  |  |  |
| --- | --- | --- | --- |
| 01 08 | Массив из восьми объектов. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 07 00 | (unsigned 16) class\_id=8 (Clock). | | |
| 09 06 00 00 01 00 00 FF | | | (octet string [6]) OBIS = 0-0:1.0.0\*255 (Clock). |
| 0F 02 | (integer 8) Attribute index=2 (date\_time). | | |
| 12 00 00 | (unsigned 16) Data index=0. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 00 01 | (unsigned 16) class\_id=1 (Data). | | |
| 09 06 01 00 60 3E 01 FF | | (octet string [6]) OBIS = 1-0:96.62.1\*255 (Status). | |
| 0F 02 | (integer 8) Attribute index=2 (value). | | |
| 12 00 00 | (unsigned 16) Data index=0. | | |

### **Пример считывания профиля загрузки (Load Profile)**

В этом примере считывание профиля загрузки выполняется в несколько шагов:

* Считывание атрибутов профиля нагрузки:

[a] список захватываемых (опрашиваемых) объектов;

[b] в период захвата (период, за который считываются данные);

[c] метод сортировки;

[d] объект сортировки;

[e] записи профиля;

[f] используемые записи.

* Считывание из буфера профиля загрузки от 2010-04-30 до текущего момента (доступ параметризируется дескриптором диапазона).
* Считывание скаляр и единиц измерения для средних значений мощности.

Считывание [a], [b], [c], [d], [e]:

|  |  |
| --- | --- |
| **Client → Server: (I frame) READ REQUEST:** | |
| 0x 7E A0 20 00 02 4A 19 41 54 E7 13 E6 E6 00 05 05 02 C0 10 02 C0 18 02 C0 20 02 C0 28 02 C0 38 AB B0 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 20 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x20 = 32 байта. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| 54 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 2; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 2. |
| E7 13 | Код целостности заголовка (HCS – Header Check Sequence). |
| E6 E6 00 | Байты логического управления каналом (LLC bytes). |
| 05 | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Запрос на чтение (Read Request). |
| 05 | Количество элементов в последовательности = 5. |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| C0 10 | Короткое имя (Short Name) = 0xC010 (объект: Load Profile; атрибут: capture\_objects) |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| C0 18 | Короткое имя (Short Name) = 0xC018 (объект: Load Profile; атрибут: capture\_period) |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| C0 20 | Короткое имя (Short Name) = 0xC020 (объект: Load Profile; атрибут: sort\_method) |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| C0 28 | Короткое имя (Short Name) = 0xC028 (объект: Load Profile; атрибут: sort\_object) |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| C0 38 | Короткое имя (Short Name) = 0xC038 (объект: Load Profile; атрибут: profile\_entries) |
| AB B0 | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE (with segmentation):** | | | |
| 0x 7E A8 8D 41 00 02 4A 19 74 25 46 E6 E7 00 0C 05 00 01 08 02 04 12 00 08 09 06 00 00 01 00 00 FF 0F 02 12 00 00 02 04 12 00 01 09 06 01 00 60 3E 01 FF 0F 02 12 00 00 02 04 12 00 03 09 06 01 00 01 05 00 FF 0F 02 12 00 00 02 04 12 00 03 09 06 01 00 02 05 00 FF 0F 02 12 00 00 02 04 12 00 03 09 06 01 00 05 05 00 FF 0F 02 12 00 00 02 04 12 00 03 09 06 01 00 06 05 00 FF 0F 02 12 00 00 02 04 12 00 03 09 06 01 00 07 05 00 FF 63 57 7E | | | |
| 7E | Флаг открытия кадра HDLC. | | |
| A8 8D | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 1, длина кадра = 0x8A=141 байт. | | |
| 41 | Адрес получателя (клиент) = 0x20 =32. | | |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | | |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | | |
| 74 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 2; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 3. | | |
| 25 46 | Код целостности заголовка (HCS – Header Check Sequence). | | |
| E6 E7 00 | Байты логического управления каналом (LLC bytes). | | |
| 0C | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Ответ на запрос чтения (Read Response). | | |
| 05 | Количество элементов в последовательности = 5. | | |
| 00 | Успешно (нет ошибок). | | |
| 01 08 | Массив из восьми объектов. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 00 08 | (unsigned 16) class\_id=8 (Clock). | | |
| 09 06 00 00 01 00 00 FF | | | (octet string [6]) OBIS = 0-0:1.0.0\*255 (Clock). |
| 0F 02 | (integer 8) Attribute index=2 (date\_time). | | |
| 12 00 00 | (unsigned 16) Data index=0. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 00 01 | (unsigned 16) class\_id=1 (Data). | | |
| 09 06 01 00 60 3E 01 FF | | (octet string [6]) OBIS = 1-0:96.62.1\*255 (Status). | |
| 0F 02 | (integer 8) Attribute index=2 (value). | | |
| 12 00 00 | (unsigned 16) Data index=0. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 07 03 | (unsigned 16) class\_id=3 (Register). | | |
| 09 06 01 00 01 05 00 FF | | | (octet string [6]) OBIS = 1-0:1.5.0\*255 (Средняя мощность +P). |
| 0F 02 | (integer 8) Attribute index=2 (value). | | |
| 12 00 00 | (unsigned 16) Data index=0. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 07 03 | (unsigned 16) class\_id=3 (Register). | | |
| 09 06 01 00 02 05 00 FF | | | (octet string [6]) OBIS = 1-0:2.5.0\*255 (Средняя мощность -P). |
| 0F 02 | (integer 8) Attribute index=2 (value). | | |
| 12 00 00 | (unsigned 16) Data index=0. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 07 03 | (unsigned 16) class\_id=3 (Register). | | |
| 09 06 01 00 05 05 00 FF | | | (octet string [6]) OBIS = 1-0:5.5.0\*255 (Средняя мощность Q1). |
| 0F 02 | (integer 8) Attribute index=2 (value). | | |
| 12 00 00 | (unsigned 16) Data index=0. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 07 03 | (unsigned 16) class\_id=3 (Register). | | |
| 09 06 01 00 06 05 00 FF | | | (octet string [6]) OBIS = 1-0:6.5.0\*255 (Средняя мощность Q2). |
| 0F 02 | (integer 8) Attribute index=2 (value). | | |
| 12 00 00 | (unsigned 16) Data index=0. | | |
| 02 04 | Структура из четырех компонентов. | | |
| 12 07 03 | (unsigned 16) class\_id=3 (Register). | | |
| 09 06 01 00 07 05 00 FF | | | (octet string [6]) OBIS = 1-0:7.5.0\*255 (Средняя мощность Q3). |
| 63 57 | Код целостности кадра HDLC (FCS – Frame Check Sequence). | | |
| 7E | Флаг закрытия кадра HDLC. | | |

|  |  |
| --- | --- |
| **Client → Server: (RR frame) RECEIVE READY:** | |
| 0x 7E A0 0A 00 02 4A 19 41 71 9C EF 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 0A | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x0A=10 байт. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| 71 | Тип кадра RR [бит 0..3] = 1; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 3. |
| 9C EF | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |  |
| --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE (end of segmentation):** | | |
| 0x 7E A0 45 41 00 02 4A 19 76 7C D7 0F 02 12 00 00 02 04 12 00 03 09 06 01 00 08 05 00 FF 0F 02 12 00 00 00 06 00 00 03 84 00 16 01 00 02 04 12 00 00 09 06 00 00 00 00 00 00 0F 00 12 00 00 00 06 00 00 39 4A 14 BA 7E | | |
| 7E | Флаг открытия кадра HDLC. | |
| A0 45 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x45=69 байт. | |
| 41 | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 76 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 3; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 3. | |
| 7C D7 | Код целостности заголовка (HCS – Header Check Sequence). | |
| 0F 02 | (integer 8) Attribute index=2 (value). | |
| 12 00 00 | (unsigned 16) Data index=0. | |
| 02 04 | Структура из четырех компонентов. | |
| 12 07 03 | (unsigned 16) class\_id=3 (Register). | |
| 09 06 01 00 08 05 00 FF | | (octet string [6]) OBIS = 1-0:8.5.0\*255 (Средняя мощность Q4). |
| 0F 02 | (integer 8) Attribute index=2 (value). | |
| 12 00 00 | (unsigned 16) Data index=0. | |
| 00 | Успешно (нет ошибок). | |
| 06 00 00 03 84 | | (unsigned 32) capture\_period = 900 = 15 sec. |
| 00 | Успешно (нет ошибок). | |
| 16 01 | | (enumerated) sort\_method = 1. |
| 00 | Успешно (нет ошибок). | |
| 02 04 | Структура из четырех компонентов. | |
| 12 00 00 | (unsigned 16) class\_id=0. | |
| 09 06 00 00 00 00 00 00 | | (octet string [6]) OBIS = 0-0:0.0.0\*0. |
| 0F 00 | (integer 8) Attribute index=0. | |
| 12 00 00 | (unsigned 16) Data index=0. | |
| 00 | Успешно (нет ошибок). | |
| 06 00 00 39 4A | | (unsigned 32) profile\_entries = 0x394A = 14666 |
| 14 BA | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | Флаг закрытия кадра HDLC. | |

Считывание [f]:

|  |  |
| --- | --- |
| **Client → Server: (I frame) READ REQUEST:** | |
| 0x 7E A0 14 00 02 4A 19 41 96 C5 4F E6 E6 00 05 01 02 C0 30 01 68 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 14 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x14 = 20 байтов. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| 96 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 3; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 4. |
| C5 4F | Код целостности заголовка (HCS – Header Check Sequence). |
| E6 E6 00 | Байты логического управления каналом (LLC bytes). |
| 05 | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Запрос на чтение (Read Request). |
| 01 | Количество элементов в последовательности = 1. |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| C0 30 | Короткое имя (Short Name) = 0xC030 (объект: Load Profile; атрибут: entries\_in\_use) |
| 01 68 | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |  |
| --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE:** | | |
| 0x 7E A0 17 41 00 02 4A 19 98 78 9C E6 E7 00 0C 01 00 06 00 00 00 6A B6 4F 7E | | |
| 7E | Флаг открытия кадра HDLC. | |
| A0 17 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x17=23 байта. | |
| 41 | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| 98 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 4; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 4. | |
| 78 9C | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E7 00 | Байты логического управления каналом (LLC bytes). | |
| 0C | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit): Ответ на запрос чтения (Read Response). | |
| 01 | Количество элементов в последовательности = 1. | |
| 00 | Успешно (нет ошибок). | |
| 06 00 00 00 6A | | (unsigned 32) profile\_entries = 0x394A = 14666. |
| B6 4F | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | Флаг закрытия кадра HDLC. | |

Считывание из буфера профиля нагрузки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Client → Server: (I frame) READ REQUEST (parameterized access):** | | | | |
| 0x 7E A0 47 00 02 4A 19 41 B8 18 5D E6 E6 00 05 01 04 C0 08 01 02 04 02 04 12 00 08 09 06 00 00 01 00 00 FF 0F 02 12 00 00 09 0C 07 DA 04 1E 05 0E 25 22 61 80 00 FF 09 0C FF FF FF FF FF FF FF FF FF 80 00 FF 01 00 C9 5F 7E | | | | |
| 7E | Флаг открытия кадра HDLC. | | | |
| A0 47 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x47 = 71 байт. | | | |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. | | | |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | | | |
| 41 | Адрес отправителя (клиент) = 0x20 =32. | | | |
| B8 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 4; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 5. | | | |
| 18 5D | Код целостности заголовка (HCS – Header Check Sequence). | | | |
| E6 E6 00 | Байты логического управления каналом (LLC bytes). | | | |
| 05 | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Запрос на чтение (Read Request). | | | |
| 01 | Количество элементов в последовательности = 1. | | | |
| 04 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[4] Parameterized access. | | | |
| C0 08 | Короткое имя (Short Name) = 0xC008 (объект: Load Profile; атрибут: buffer) | | | |
| 01 | Selector = [2] range\_descriptor. | | | |
| 02 04 | Структура из четырех компонентов. | | | |
| 02 04 | Структура из четырех элементов (Capture\_object\_definition): | | | |
| 12 00 08 | (unsigned 16) class\_id=8 (Clock). | | | |
| 09 06 00 00 01 00 00 FF | | (octet string [6]) OBIS = 0-0:1.0.0\*255 (Clock). | | |
| 0F 02 | (integer 8) Attribute index=2 (date\_time). | | | |
| 12 00 00 | (unsigned 16) Data index=0. | | | |
| 09 0C 07 DA 04 1E 05 0E 25 22 61 80 00 FF | | | | (octet string [12]) from 2010-04-30 14:37:34 |
| 09 0C FF FF FF FF FF FF FF FF FF 80 00 FF | | | (octet string [12]) до текущего момента | |
| 01 00 | Массив из 0 элементов (пустой массив), нет Selected\_values | | | |
| C9 5F | Код целостности кадра HDLC (FCS – Frame Check Sequence). | | | |
| 7E | Флаг закрытия кадра HDLC. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE (with segmentation):** | | | | |
| 0x 7E A8 81 41 00 02 4A 19 BA CD D8 E6 E7 00 0C 01 00 01 0E 02 08 09 0C 07 DA 04 1E 05 0E 2D 00 FF 80 00 00 09 03 80 00 00 06 00 00 00 5C 06 00 00 00 00 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C 02 08 09 0C 07 DA 04 1E 05 0F 00 00 FF 80 00 00 00 06 00 00 00 58 00 06 00 00 00 58 06 00 00 00 58 06 00 00 00 58 06 00 00 00 58 02 08 00 00 06 00 00 00 5C 00 06 00 00 00 5C 4A 4B 7E | | | | |
| 7E | | Флаг открытия кадра HDLC. | | |
| A8 81 | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 1, длина кадра = 0x81=129 байтов. | | |
| 41 | | Адрес получателя (клиент) = 0x20 =32. | | |
| 00 02 | | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | | |
| 4A 19 | | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | | |
| ВА | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 5; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 5. | | |
| CD D8 | | Код целостности заголовка (HCS – Header Check Sequence). | | |
| E6 E7 00 | | Байты логического управления каналом (LLC bytes). | | |
| 0C | | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Ответ на запрос чтения (Read Response). | | |
| 01 | | Количество элементов в последовательности = 1. | | |
| 00 | | Успешно (нет ошибок). | | |
| 01 0E | | Массив из 14 элементов | | |
| 02 08 | | Структура из 8 компонентов | | |
| 09 0C 07 DA 04 1E 05 0E 2D 00 FF 80 00 00 | | | | (octet string [12]) Метка времени 2010-04-30 14:45:00. |
| 09 03 80 00 00 | | | (octet string [3]) Статус = 0x800000 | |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность +P = 0x0000005C = 92 | |
| 06 00 00 00 00 | | | (unsigned 32) Средняя мощность -P = 0x00000000 = 0 | |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q1 = 0x0000005C = 92 | |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q2 = 0x0000005C = 92 | |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q3 = 0x0000005C = 92 | |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q4 = 0x0000005C = 92 | |
| 02 08 | | Структура из 8 компонентов | | |
| 09 0C 07 DA 04 1E 05 0F 00 00 FF 80 00 00 | | | | (octet string [12]) Метка времени 2010-04-30 15:00:00. |
| 00 | | | Статус = тоже значение, что и в предыдущей записи | |
| 06 00 00 00 58 | | | (unsigned 32) Средняя мощность +P = 0x00000058 = 88 | |
| 00 | | | Средняя мощность -P = значение, что и в предыдущей записи | |
| 06 00 00 00 58 | | | (unsigned 32) Средняя мощность Q1 = 0x00000058 = 88 | |
| 06 00 00 00 58 | | | (unsigned 32) Средняя мощность Q2 = 0x00000058 = 88 | |
| 06 00 00 00 58 | | | (unsigned 32) Средняя мощность Q3 = 0x00000058 = 88 | |
| 06 00 00 00 58 | | | (unsigned 32) Средняя мощность Q4 = 0x00000058 = 88 | |
| 02 08 | | Структура из 8 компонентов | | |
| 00 | Метка времени = тоже значение, что и в предыдущей записи + период захвата. | | | |
| 00 | | | Статус = тоже значение, что и в предыдущей записи | |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность +P = 0x0000005C = 92 | |
| 00 | | | Средняя мощность -P = значение, что и в предыдущей записи | |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q1 = 0x0000005C = 92 | |
| 4A 4B | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | | |
| 7E | | Флаг закрытия кадра HDLC. | | |

|  |  |
| --- | --- |
| **Client → Server: (RR frame) RECEIVE READY:** | |
| 0x 7E A0 0A 00 02 4A 19 41 D1 96 4A 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 0A | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x0A=10 байт. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| D1 | Тип кадра RR [бит 0..3] = 1; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 6. |
| 96 4A | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |
| --- |
| **Server → Client: (I frame) READ REASPONSE (with segmentation):** |
| 0x 7E A8 7D 41 00 02 4A 19 BC 30 9B 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C 02 08 00 00 06 00 00 00 58 00 06 00 00 00 58 06 00 00 00 58 06 00 00 00 58 06 00 00 00 58 02 08 00 00 06 00 00 00 5C 00 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C 02 08 09 0C 07 DA 04 1E 05 10 00 00 FF 80 00 00 00 06 00 00 00 58 00 06 00 00 00 58 06 00 00 00 58 06 00 00 00 58 99 53 7E |

|  |
| --- |
| **Client → Server: (RR frame) RECEIVE READY:** |
| 0x 7E A0 0A 00 02 4A 19 41 F1 94 6B 7E |

|  |
| --- |
| **Server → Client: (I frame) READ REASPONSE (with segmentation):** |
| 0x 7E A8 80 41 00 02 4A 19 BE 3C 01 06 00 00 00 58 02 08 09 0C 07 DA 04 1E 05 10 04 00 FF 80 00 00 09 03 00 00 94 06 00 00 00 18 00 06 00 00 00 18 06 00 00 00 18 06 00 00 00 18 06 00 00 00 18 02 08 09 0C 07 DA 05 03 01 0D 10 0E FF 80 00 00 09 03 08 00 40 00 00 00 00 00 00 02 08 09 0C 07 DA 05 03 01 0D 16 23 FF 80 00 00 09 03 00 00 14 06 00 00 00 24 00 06 00 00 00 24 06 00 00 00 24 D2 A3 7E |

|  |
| --- |
| **Client → Server: (RR frame) RECEIVE READY:** |
| 0x 7E A0 0A 00 02 4A 19 41 11 9A 8C 7E |

|  |
| --- |
| **Server → Client: (I frame) READ REASPONSE (with segmentation):** |
| 0x 7E A8 7D 41 00 02 4A 19 B0 5C 51 06 00 00 00 24 06 00 00 00 24 02 08 09 0C 07 DA 05 03 01 0D 1E 00 FF 80 00 00 09 03 80 00 C0 06 00 00 00 20 00 06 00 00 00 20 06 00 00 00 20 06 00 00 00 20 06 00 00 00 20 02 08 00 00 06 00 00 00 58 00 06 00 00 00 58 06 00 00 00 58 06 00 00 00 58 06 00 00 00 58 02 08 09 0C 07 DA 05 03 01 0E 00 00 FF 80 00 00 09 03 80 00 00 06 00 00 00 5C FC FF 7E |

|  |
| --- |
| **Client → Server: (RR frame) RECEIVE READY:** |
| 0x 7E A0 0A 00 02 4A 19 41 31 98 AD 7E |

|  |  |  |  |
| --- | --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE (end of segmentation):** | | | |
| 0x 7E A0 65 41 00 02 4A 19 B2 D7 34 00 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C 02 08 00 09 03 80 00 C0 06 00 00 00 4C 00 06 00 00 00 4C 06 00 00 00 4C 06 00 00 00 4C 06 00 00 00 4C 02 08 00 09 03 80 00 00 06 00 00 00 5C 00 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C 06 00 00 00 5C BA 3A 7E | | | |
| 7E | | Флаг открытия кадра HDLC. | |
| A0 65 | | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x81=129 байтов. | |
| 41 | | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| В2 | | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 1; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 5. | |
| D7 34 | | Код целостности заголовка (HCS – Header Check Sequence). | |
| 00 | | | Средняя мощность -P = значение, что и в предыдущей записи |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q1 = 0x0000005C = 92 |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q2 = 0x0000005C = 92 |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q3 = 0x0000005C = 92 |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q4 = 0x0000005C = 92 |
| 02 08 | | Структура из 8 компонентов | |
| 00 | Метка времени = тоже значение, что и в предыдущей записи + период захвата. | | |
| 09 03 80 00 C0 | | | (octet string [3]) Статус = 0x8000C0 |
| 06 00 00 00 4C | | | (unsigned 32) Средняя мощность +P = 0x0000004C = 76 |
| 00 | | | Средняя мощность -P = значение, что и в предыдущей записи |
| 06 00 00 00 4C | | | (unsigned 32) Средняя мощность Q1 = 0x0000004C = 76 |
| 06 00 00 00 4C | | | (unsigned 32) Средняя мощность Q2 = 0x0000004C = 76 |
| 06 00 00 00 4C | | | (unsigned 32) Средняя мощность Q3 = 0x0000004C = 76 |
| 06 00 00 00 4C | | | (unsigned 32) Средняя мощность Q4 = 0x0000004C = 76 |
| 02 08 | | Структура из 8 компонентов | |
| 00 | Метка времени = тоже значение, что и в предыдущей записи + период захвата. | | |
| 09 03 80 00 00 | | | (octet string [3]) Статус = 0x800000 |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность +P = 0x0000005C = 92 |
| 00 | | | Средняя мощность -P = значение, что и в предыдущей записи |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q1 = 0x0000005C = 92 |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q2 = 0x0000005C = 92 |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q3 = 0x0000005C = 92 |
| 06 00 00 00 5C | | | (unsigned 32) Средняя мощность Q4 = 0x0000005C = 92 |
| BA 3A | | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | | Флаг закрытия кадра HDLC. | |

Считывание скаляр и единиц измерения

|  |  |
| --- | --- |
| **Client → Server: (I frame) READ REQUEST:** | |
| 0x 7E A0 14 00 02 4A 19 41 5A A5 43 E6 E6 00 05 01 02 36 10 DB 61 7E | |
| 7E | Флаг открытия кадра HDLC. |
| A0 14 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x14 = 20 байтов. |
| 00 02 | Адрес получателя (сервер) верхняя часть HDLC адреса = 0x0001. |
| 4A 19 | Адрес получателя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. |
| 41 | Адрес отправителя (клиент) = 0x20 =32. |
| 5A | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 5; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 2. |
| A5 43 | Код целостности заголовка (HCS – Header Check Sequence). |
| E6 E6 00 | Байты логического управления каналом (LLC bytes). |
| 05 | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Запрос на чтение (Read Request). |
| 01 | Количество элементов в последовательности = 1. |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| 36 10 | Короткое имя (Short Name) = 0x3610 (объект: Средняя мощность +P; атрибут: scaler\_unit) |
| DB 61 | Код целостности кадра HDLC (FCS – Frame Check Sequence). |
| 7E | Флаг закрытия кадра HDLC. |

|  |  |  |
| --- | --- | --- |
| **Server → Client: (I frame) READ REASPONSE:** | | |
| 0x 7E A0 18 41 00 02 4A 19 D4 E4 0D E6 E7 00 0C 01 00 02 02 0F 00 16 1B 61 63 7E | | |
| 7E | Флаг открытия кадра HDLC. | |
| A0 18 | Тип кадра [биты 12..15] = 0xA, сегментация [бит 11] = 0, длина кадра = 0x18=24 байта. | |
| 41 | Адрес получателя (клиент) = 0x20 =32. | |
| 00 02 | Адрес отправителя (сервер) верхняя часть HDLC адреса = 0x0001. | |
| 4A 19 | Адрес отправителя (сервер) нижняя часть HDLC адреса = 0x128C = 4748. | |
| D4 | Тип кадра I [бит 0] = 0; счетчик отправленных кадров (SendNr): [биты 1..3] = 2; бит опроса/завершения (PF): [бит 4] = 1; счетчик принятых кадров (RecNr): [биты 5..7] = 6. | |
| E4 0D | Код целостности заголовка (HCS – Header Check Sequence). | |
| E6 E7 00 | Байты логического управления каналом (LLC bytes). | |
| 0C | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): Ответ на запрос чтения (Read Response). | |
| 01 | Количество элементов в последовательности = 1. | |
| 00 | Успешно (нет ошибок). | |
| 02 02 | Структура из двух компонентов | |
| 0F 00 | | (integer 8) Величина = 0 |
| 16 1B | | (enumerated) Единицы измерения = 0x1B = 27 (watts [W]) |
| 61 63 | Код целостности кадра HDLC (FCS – Frame Check Sequence). | |
| 7E | Флаг закрытия кадра HDLC. | |

### **Биты статуса в профиле нагрузки**

Биты статуса в профиле загрузки имеют следующие значения:

|  |  |
| --- | --- |
| **№ Бита** | **Описание** |
| 23 | Нормальное завершение периода интеграции на основе RTC растра [STC\_EOI\_REG\_INT] |
| 22 | Не используется [STC\_EOI\_REG\_EXT] |
| 21 | Зарезервировано для использования в будущем |
| 20 | Не используется [STC\_EOI] |
| 19 | Не используется [STC\_SOI] |
| 18 | Зарезервировано для использования в будущем |
| 17 | Зарезервировано для использования в будущем |
| 16 | Зарезервировано для использования в будущем |
| 15 | Не используется [STC\_CLKADJ\_BEFORE] |
| 14 | Очищен профиль загрузки [STC\_LP\_CLEAR] |
| 13 | Не используется [STC\_EVP\_CLEAR] |
| 12 | Зарезервировано для использования в будущем |
| 11 | Зарезервировано для использования в будущем |
| 10 | Зарезервировано для использования в будущем |
| 09 | Зарезервировано для использования в будущем |
| 08 | Зарезервировано для использования в будущем |
| 07 | Падение мощности [STC\_PWR\_DOWN] |
| 06 | Скачек мощности [STC\_PWR\_UP] |
| 05 | После регулировки часов [STC\_CLKADJ\_AFTER] |
| 04 | Расчетный период сброшен [STC\_BP\_RESET] |
| 03 | Летнее время [STC\_SEASON] |
| 02 | Искажен период интеграции [STC\_BADPERIOD] |
| 01 | Отказ часов [STC\_CLOCK\_FAILURE] |
| 00 | Фатальная ошибка [STC\_ERROR] |

*Для примера, значение статуса = 0x800000 значит, что нормально завершился период интеграции на основе RTC растра.*

## **Профиль тарификации**

Профиль тарификации моделируется интерфейсным классом «Profile generic» (см. раздел 3.1.1).

|  |  |  |
| --- | --- | --- |
| **OBIS** | **Base name** | **Interface Class (IC), Object description** |
| 1-0:98.1.0\*126 | 0xC100 | (IC: Profile Generic) Billing profile |

**Замечание**: *Логическое имя (код OBIS) однозначно определяет смысл информации, представленной объектом, независимым от производителя образом. Фактические значения коротких имен (базовые, основные имена) зависят от количества и типа объектов, созданных в устройстве и стратегии ассоциирования, используемой производителем, поэтому они могут быть разными в различных устройствах.*

### **Пример считывания профиля тарификации (Billing profile)**

В этом примере считывание профиля тарификации выполняется в несколько шагов:

* Считывание атрибутов профиля тарификации:

[a] список захватываемых (опрашиваемых) объектов;

[b] период захвата (период, за который считываются данные);

[c] метод сортировки;

[d] объект сортировки;

[e] записи профиля;

[f] используемые записи.

* Считывание из буфера профиля тарификации.

## **Параметры**

### **Команда синхронизации часов ±59 секунд**

Логическое имя параметра OBIS = 1-0:96.130.3\*255 (*1.0.96.130.3.255*).

Короткое имя параметра [SN\_OLD\_TT] = 0xD900.

Формат значения параметра – два ASCII символа, обозначающие шестнадцатеричное число:

* [0x01…0x3B] – регулировка часов вперед на [+1…+59] секунды;
* [0xFF…0xC5] – регулировка часов назад на [-1…-59] секунды.

Пример:

|  |  |
| --- | --- |
| **Client → Server: (I frame) Write Request:** | |
| 0x 7E A0 19 00 02 A0 D7 65 72 56 F6 E6 E6 00 *06 01 02 D9 08* 01 0A 02 **33 42** E4 2A 7E | |
| **06** | Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit ): **Write Request** (запрос на запись). |
| 01 | Количество элементов в последовательности = 1. |
| 02 | Определение переменной доступа (VariableAccessSpecification) = CHOICE[2] переменная name. |
| D9 08 | Короткое имя (Short Name) = 0xD908 (пишем атрибут: value объекта 0xD900) |
| 01 | Количество параметров = 1. |
| 0A 02 | Отображаемая строка размером 2 байта (visible string (size(2))). |
| **33 42** | 0x3342 переводится в ASCII как 3B. 0x3B = +59 секунд |

|  |  |
| --- | --- |
| **Server → Client: (WriteResponse):** | |
| 0x 7E A0 13 65 00 02 A0 D7 12 F1 67 E6 E7 00 *0D 01* 01 00 FB 9D 7E | |
| **0D** | **WriteResponse** (ответ на запрос записи параметра). |
| 01 | Количество элементов в последовательности. |
| 01 | Количество установленных (записанных) параметров. |
| 00 | Успешно. |

# **Описание интерфейсных классов**

## **COSEM Interface Class: Register (class\_id:3)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Register (Class\_id=3)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. value 3. scaler\_unit | Static  Dynamic  Static | Octet-string  CHOICE  scaler\_unit\_type | X  X+0x08  X+0x10 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. reset (data) | | Необязательный | X+0x28 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Register» |
| value | Содержит текущие значения  CHOICE  {  *-- simple data types*  null-data [0],  bit-string [4],  double-long [5],  double-long-unsigned [6],  octet-string [9],  visible-string [10],  utf8-string [12],  integer [15],  long [16],  unsigned [17],  long-unsigned [18],  long64 [20],  long64-unsigned [21],  enum [22],  float32 [23],  float64 [24].  } |
| scaler\_unit | Содержит информацию об единице измерения и множителе.  scaler\_unit\_type::structure  {  scaler,  unit  }  scaler: integer  Степень числа 10. Значение value должно быть умножено на 10 в степени scaler.  unit: enum  Единица измерения |
| **Методы** | |
| reset (data) | Вызов этого метода устанавливает значение value в состояние «по умолчанию». |

## **COSEM Interface Class: Extended register (class\_id:4)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Extended register (Class\_id=4)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. value 3. scaler\_unit 4. status 5. capture\_time | Static  Dynamic  Static  Dynamic  Dynamic | Octet-string  CHOICE  scaler\_unit\_type  CHOICE  Octet-string | X  X+0x08  X+0x10  X+0x18  X+0x20 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. reset (data) | | Необязательный | X+0x38 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Extended register» |
| value | Содержит текущие значения  CHOICE  {  *-- simple data types*  null-data [0],  bit-string [4],  double-long [5],  double-long-unsigned [6],  octet-string [9],  visible-string [10],  utf8-string [12],  integer [15],  long [16],  unsigned [17],  long-unsigned [18],  long64 [20],  long64-unsigned [21],  enum [22],  float32 [23],  float64 [24].  } |
| scaler\_unit | Содержит информацию об единице измерения и множителе.  scaler\_unit\_type::structure  {  scaler,  unit  }  scaler: integer  Степень числа 10. Значение value должно быть умножено на 10 в степени scaler.  unit: enum  Единица измерения |
| status | Предоставляет сведения о состоянии объекта типа «Расширенный регистр». Для каждого экземпляра объекта должен быть предусмотрен подходящий по смыслу элемент статуса.  CHOICE  {  *-- simple data types*  null-data [0],  bit-string [4],  double-long-unsigned [6],  octet-string [9],  visible-string [10],  utf8-string [12],  unsigned [17],  long-unsigned [18],  long64-unsigned [21].  }  Тип данных и кодировка зависит от экземпляра и выбора производитель. Для интерпретации, может потребоваться дополнительная информация от производителя.  Определение: выбор зависит от определения типа состояния. |
| capture\_time | Предоставляет сведения о дате и времени объекта типа «Расширенный регистр», показывает, когда значение атрибута было захвачено.  Байтовая строка, отформатированная, как указано в 4.6.1 для даты-времени. |
| **Методы** | |
| reset (data) | Этот метод принудительно сбрасывает объект.  Вызов этого метода устанавливает значение value в состояние «по умолчанию». Значение «по умолчанию» – определенная для экземпляра объекта константа (постоянная величина).  Атрибут capture\_time устанавливается в значение времени на момент выполнения сброса.  data:: = integer(0) |

## **COSEM Interface Class: Demand register (class\_id:5)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Demand register (Class\_id=5)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. current\_average\_value 3. last\_average\_value 4. scaler\_unit 5. status 6. capture\_time 7. start\_time\_current 8. period 9. number\_of\_periods | Static  Dynamic  Dynamic  Static  Dynamic  Dynamic  Dynamic  Static  Static | Octet-string  CHOICE  CHOICE  scaler\_unit\_type  CHOICE  Octet-string  Octet-string  double-long-unsigned  long-unsigned | X  X+0x08  X+0x10  X+0x18  X+0x20  X+0x28  X+0x30  X+0x38  X+0x40 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. reset (data) 2. next\_period (data) | | Необязательный  Необязательный | X+0x48  X+0x50 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Demand register» |
| current\_average\_value | Предоставляет текущее значение (текущее потребление) энергии, накопленное с момента start\_time, деленное на number\_of\_periods\*period.  Тип данных значения зависит от экземпляра объекта, определяется заданным logical\_name и задается производителем. Тип должен быть выбран так, чтобы вместе с logical\_name была возможна однозначная интерпретация значения.  Если измеряемая величина, отлична от энергии, то могут применяться другие методы расчета (например, для расчета средних или текущих значений напряжения).  CHOICE (перечисление типов данных) совпадает с приведенным в описании IC «Register». |
| last\_average\_value | Предоставляет значение накопленной энергии (за последние number\_of\_periods\*period) делится на number\_of\_periods\*period. Энергия текущего (не завершенного) периода не учитывается при расчете.  Если измеряемая величина, отлична от энергии, то могут применяться другие методы расчета (например, для расчета средних или текущих значений напряжения).  CHOICE (перечисление типов данных) совпадает с приведенным в описании IC «Register». |
| scaler\_unit | Содержит информацию об единице измерения и множителе.  См. описание, приведенное в IC «Register». |
| status | Предоставляет сведения о состоянии объекта типа «Регистр потребления». Для каждого экземпляра объекта должен быть предусмотрен подходящий по смыслу элемент статуса.  CHOICE (перечисление типов данных) совпадает с приведенным в описании IC «Extended register».  Определение: выбор зависит от определения типа состояния. |
| capture\_time | Предоставляет дату и время, когда значение last\_average\_value было рассчитанно.  Байтовая строка, отформатированная, как указано в 4.6.1 для даты-времени (date-time). |
| start\_time\_current | Предоставляет дату и время, когда началось измерение значения current\_average\_value.  Байтовая строка, отформатированная, как указано в 4.6.1 для даты-времени (date-time). |
| period | Интервал между двумя последовательными обновлениями значения last\_average\_value. (number\_of\_periods\*period знаменатель при расчете потребления).  double-long-unsigned, период измерения в секундах.  Поведение счетчика после записи нового значения этого атрибута определяется изготовителем. |
| number\_of\_periods | Число периодов, используется для вычисления значения last\_average\_value.  number\_of\_periods >= 1.  – number\_of\_periods > 1 указывает, что значение last\_average\_value представляет «протяженное потребление» ("sliding demand")  – number\_of\_periods = 1 означает, что значение last\_average\_value представляет собой «ограниченное потребление» ("block demand").  Поведение счетчика после записи нового значения этого атрибута определяется изготовителем. |
| **Методы** | |
| reset (data) | Этот метод принудительно сбрасывает объект.  Вызов этого метода провоцирует следующие действия:  - текущий период (current period) прекращается;  – current\_average\_value и last\_average\_value устанавливаются в  их значения «по умолчанию»;  – capture\_time и start\_time\_current устанавливаются в значение равное времени выполнения метода reset (data).  data:: = integer(0) |
| next\_period (data) | Этот метод используется для запуска нормального (штатного) завершение (и перезапуска) период.  Закрывает (сбрасывает) текущий измерительный период.  Обновляет capture\_time и start\_time и копирует current\_average\_value в last\_average\_value, задает current\_average\_value значение по умолчанию.  Запускает следующий период измерения.  Примечание: *Старое значение last\_average\_value (и capture\_time) может быть прочитано во время периода измерения ("period"). Старое значение current\_average\_value больше не доступно через интерфейс*. |

## **COSEM Interface Class: Clock (class\_id:8)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Clock (Class\_id=8)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. time 3. time\_zone 4. status 5. daylights\_savings\_begin 6. daylights\_savings\_end 7. daylights\_savings\_deviation 8. daylights\_savings\_enabled 9. clock\_base | Static  Dynamic  Static  Dynamic  Static  Static  Static  Static  Static | Octet-string [6]  Octet-string [12]  Long  Unsigned  Octet-string [12]  Octet-string [12]  integer  boolean  enum | X  X+0x08  X+0x10  X+0x18  X+0x20  X+0x28  X+0x30  X+0x38  X+0x40 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. adjust\_to\_quarter (data) 2. adjust\_to\_measuring\_period (data) 3. adjust\_to\_minute (data) 4. adjust\_to\_preset\_time 5. preset\_adjusting\_time (data) 6. shift\_time (data) | | Необязательный  Необязательный  Необязательный  Необязательный  Необязательный  Необязательный | X+0x60  X+0x68  X+0x70  X+0x78  X+0x80  X+0x88 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Clock» |
| time | Cодержит локальную дату и время счетчика, ее отклонение по UTC и статус.  Октет-строка, отформатированная как указано в 4.1.6.1 (*EXCERPT DLMS UA Blue Book: COSEM interface classes and OBIS identification system*) для даты-времени:  OCTET STRING (SIZE(12))  {  year highbyte,  year lowbyte,  month,  day of month,  day of week,  hour,  minute,  second,  hundredths of second,  deviation highbyte,  deviation lowbyte,  clock status  }  Когда этот атрибут задан, все поля в октет-строке, представляющей дату-время должны быть заполнены и местные дата и время в счетчике должны быть установлены в соответствии с определенными правилами.  Меняются только заданные поля даты-времени.  *Пример: для установки даты без изменения времени, для всех соответствующих времени октетов в октетной строке, представляющей дату и время должно быть установлено значение «не уточняется».* |
| time\_zone | Отклонение местного времени от нормального времени по Гринвичу в минутах. Величина зависит от географического расположения счетчика. |
| status | Статус часов поддерживается счетчиком. Элемент статус часов указан в атрибуте времени и равен значению этого атрибута.  Без знака, форматируется как указано для статуса часов. |
| daylight\_savings\_begin | Определяет местное изменение даты и времени, когда местное время начинает отклоняться от нормального времени.  Для общих определений заданных шаблонов.  Октетная строка, отформатированная как указано для даты и времени. |
| daylight\_savings\_end | Определяет местное изменение даты и времени, когда местное время заканчивает отклоняться от нормального времени.  Октетная строка, отформатированная как указано для даты и времени. |
| daylights\_savings\_deviation | Содержит количество минут, на которое обобщенное отклонение времени будет корректироваться при переходе на летнее время.  Целое число: отклонение в диапазоне ± 120 мин. |
| daylights\_savings\_enabled | Логическое значение:  TRUE = DST включено,  FALSE = DST выключено.  Примечание: этот параметр для включения и отключения функции перехода на летнее время. Текущее состояние отображается в атрибуте «статус». |
| clock\_base | Определяет от куда пришла основная (базовая) информация о времени.  Перечисление:  (0) не определено,  (1) внутренний кристалл,  (2) питающая сеть с частотой 50 Гц,  (3) питающая сеть с частотой 60 Гц,  (4) GPS (глобальная система позиционирования),  (5) управляется по радио |
| **Методы** | |
| adjust\_to\_quarter (data) | Устанавливает время счетчика до значения ближайшего (+/-) четверти часа.  (\*:00, \*:15, \*:30, \*:45).  data ::= integer (0) |
| adjust\_to\_measuring\_period (data) | Устанавливает время счетчика до значения ближайшего (+/-) к точке начала периода измерения.  data ::= integer (0) |
| adjust\_to\_minute (data) | Устанавливает время счетчика до ближайшей минуты.  Если second\_counter < 30 сек, second\_counter устанавливается в значение 0.  Если second\_counter >= 30 сек, second\_counter устанавливается в значение 0, а minute\_counter и все зависимые значения часов увеличиваются на единицу, если это необходимо.  data ::= integer (0) |
| adjust\_to\_preset\_time | Этот метод используется в сочетании с методом preset\_adjusting\_time. Если время счетчика лежит между validity\_interval\_start и validity\_interval\_end, то время устанавливается в значение preset\_time.  data ::= integer (0) |
| preset\_adjusting\_time (data) | Устанавливает время в новое значение (preset\_time) и определяет validity\_interval, в течение которого новое время может быть активировано.  data ::= structure  {  preset\_time: octet-string,  validity\_interval\_start: octet-string,  validity\_interval\_end: octet-string  }  Все октет-строки форматируются, как указано для даты и времени. |
| shift\_time (data) | Сдвигает время на n (-900 <= n <= 900) с.  data ::= long |

## **COSEM Interface Class: Schedule (class\_id:10)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Schedule (Class\_id=10)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. entries | Static  Static | octet-string  array | X  X+0x08 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. enable/disable (data) 2. insert (data) 3. delete (data) | | Необязательный  Необязательный  Необязательный | X+0x20  X+0x28  X+0x30 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Schedule» |
| entries | Определяет сценарии, которые будут выполняться в определенное время. Для каждой записи можно выполнить только один сценарий.  array schedule\_table\_entry  schedule\_table\_entry::= structure  {  index: long-unsigned,  enable: boolean,  script\_logical\_name: octet-string,  script\_selector: long-unsigned,  switch\_time: octet-string,  validity\_window: long-unsigned,  exec\_weekdays: bit-string,  exec\_specdays: bit-string,  begin\_date: octet-string,  end\_date: octet-string  }  Где:  – script\_logical\_name: определяет логическое имя объекта "Script table";  – script\_selector: определяет script\_identifier для выполняемого скрипта;  – switch\_time принимает подстановочные знаки для определения повторяющихся записей. Формат octet-string соответствует правилам, установленным в 4.1.6.1 для time (времени);  – validity\_window определяет период в минутах, в течение которого запись будет обработана после сбоя питания. (время между определенным switch\_time и фактическим power\_up) 0xFFFF: сценарий обрабатывается в любое время;  – exec\_weekdays определяет дни недели, в которые запись действительна;  – exec\_specdays выполняет ссылку на IC "Special days table", day\_id;  – start\_date и end\_date определяют промежуток времени (период), в течение которого запись действительна (разрешены подстановочные знаки). Формат соответствует правилам, установленным в 4.6.1 для date (даты). |
| **Методы** | |
| enable/disable (data) | Устанавливает в True запрещающий бит для диапазона записей A и затем разрешает записи диапазона B.  data::= structure  {  firstIndexA: long-unsigned,  lastIndexA: long-unsigned,  firstIndexB: long-unsigned,  lastIndexB: long-unsigned  }  Где:  – firstIndexA: первый индекс запрещенного диапазона,  – lastIndexA: последний индекс запрещенного диапазона,  – firstIndexB: первый индекс разрешенного диапазона,  – lastIndexB: последний индекс разрешенного диапазона,  firstIndexA/B < lastIndexA/B: все записи диапазона A/B запрещены/разрешены (disabled/enabled),  firstIndexA/B = lastIndexA/B: одна запись запрещена/разрешена (disabled/enabled),  firstIndexA/B > lastIndexA/B: ничего не запрещено/разрешено,  firstIndexA/B and lastIndexA/B > 9999: нет запрещенных/разрешенных записей |
| insert (data) | Вставляет новую запись в таблицу. Если запись с таким индексом уже существует, то существующая запись будет заменена новой записью (перезаписана).  entry: schedule\_table\_entry  data::= соответствующая запись. |
| delete (data) | Удаляет диапазон записей из таблицы.  data::= structure  {  firstIndex: long-unsigned,  lastIndex: long-unsigned  }  Где:  – firstIndex: первый индекс удаляемого диапазона,  – lastIndex: последний индекс удаляемого диапазона,  firstIndex < lastIndex: все записи диапазона A/B будут удалены,  firstIndex = lastIndex: одна запись будет удалена,  firstIndex > lastIndex: ничего не удаляется |

## **COSEM Interface Class: Special days table (class\_id:11)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Special days table (Class\_id=11)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. entries | Static  Static | octet-string  array | X  X+0x08 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. insert (data) 2. delete (data) | | Необязательный  Необязательный | X+0x10  X+0x18 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Special days table» |
| entries | Определяет (указывает) специальный идентификатор дня для заданной даты. Дата может иметь подстановочные знаки для повторяющихся особых дней, как Рождество.  array spec\_day\_entry  spec\_day\_entry::= structure  {  index: long-unsigned,  specialday\_date: octet-string,  day\_id: unsigned  }  Где:  - specialday\_date форматируется в соответствии с правилами, установленными в 4.6.1 для даты;  - диапазон day\_id должен соответствовать длине битовой строки exec\_specdays в объекте, связанном с IC «Schedule». |
| **Методы** | |
| insert (data) | Вставляет новую запись в таблицу.  entry::= spec\_day\_entry  Если вставляется специальный день с индексом или датой, совпадающими с уже заданным днем, старая запись будет перезаписана. |
| delete (data) | Удаляет запись из таблице.  index – индекс записи, которая должна быть удалена.  data::= long-unsigned |

## **COSEM Interface Class: Image transfer (class\_id:18)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Image transfer (Class\_id=18)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. image\_block\_size 3. image\_transferred\_blocks\_status 4. image\_first\_not\_transferred\_block\_number 5. image\_transfer\_enabled 6. image\_transfer\_status 7. image\_to\_activate\_info | Static  Static  Dynamic  Dynamic  Static  Dynamic  Dynamic | octet-string  double-long-unsigned  bit-string  double-long-unsigned  boolean  enum  array | X  X+0x08  X+0x10  X+0x18  X+0x20  X+0x28  X+0x30 |
| **Определенные методы** | | **Обязательный/ Необязательный** |  |
| 1. image\_transfer\_initiate 2. image\_block\_transfer 3. image\_verify 4. image\_activate | | Обязательный  Обязательный  Обязательный  Обязательный | X+0x40  X+0x48  X+0x50  X+0x58 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Image transfer» |
| image\_block\_size | Содержит ImageBlockSize, выраженный в октетах, который может быть обработан сервером. ImageBlockSize не должен превышать согласованный (заданный) ServerMaxReceivePduSize.  Примечание 1: *image\_block\_size является свойством сервера.* |
| image\_transferred\_blocks\_status | Предоставляет сведения о состоянии передачи каждого ImageBlock.  Каждый бит в битовой строке содержит (предоставляет) информацию об одном единственном ImageBlock:  0 = Not transferred (не передан),  1 = Transferred (передан).  Примечание 2: размер атрибута может быть динамическим, т.е. он может расти при получении новых ImageBlocks. |
| image\_first\_not\_transferred\_block\_number | Предоставляет ImageBlockNumber первого не перенесенного ImageBlock. После того, как передача image (образ) будет завершена, возвращаемое значение должно быть равно или больше количества блоков рассчитанное по размеру образа и ImageBlockSize. |
| image\_transfer\_enabled | Управляет включением процесса передачи изображений (образов, images).  boolean: false = отключено,  true = включено  Методы передачи изображения могут быть успешно вызваны только в том случае, если значение этого атрибута true. Установка значения этого атрибута в false отключает все методы (вызвать их не удастся). |
| image\_transfer\_status | Хранит состояние процесса передачи изображений.  enum:  (0) Передача образа не инициирована,  (1) Передача образа инициирована,  (2) Начата проверка образа,  (3) Проверка образа успешно завершена,  (4) Ошибка проверки образа,  (5) Инициирована активация образа,  (6) Активация образа успешно завершена,  (7) Активировать образ не удалось. |
| image\_to\_activate\_info | Предоставляет информацию о готовности образа к активации. Она создается по результату процесса проверки образа. Клиент может проверить эту информацию перед активацией образа.  array image\_to\_activate\_info\_element  image\_to\_activate\_info\_element::= structure  {  image\_to\_activate\_size: double-long-unsigned,  image\_to\_activate\_identification: octet-string,  image\_to\_activate\_signature: octet-string  }  Где:  – image\_to\_activate\_size – размер образа, который будет активирован, выраженный в октетах;  – image\_to\_activate\_identification является идентификатором образа, который должен быть активирован, и может содержать информацию о производителе, типе устройства, информацию о версии и т.д.;  – image\_to\_activate\_signature – подпись образа, который должен быть активирован.  Примечание 3: *алгоритм генерации подписи выходит за рамки этой спецификации.* |
| **Методы** | |
| image\_transfer\_initiate | Инициирует процесс передачи образа.  data::= structure  {  image\_identifier: octet-string,  image\_size: double-long-unsigned  }  Где:  – image\_identifier идентифицирует образ, который будет передаваться;  – image\_size содержит ImageSize, выраженный в октетах.  Примечание 4: *image\_identifier идентифицирует образ, который будет передаваться (контейнер), но это не обязательно связано с его содержимым, т.е. образами, которые будут активированы. Эта информация может быть извлечена из атрибута image\_to\_activate после проверки переданного образа.*  После успешного вызова метода атрибуту image\_transfer\_status присваивается значение (1), а атрибуту image\_first\_not\_transferred\_block\_number присваивается значение 0. Любой последующий вызов метода сбрасывает весь процесс передачи образа и все ImageBlocks должны быть переданы снова. |
| image\_block\_transfer | Перенос (передача) одного блока образа на сервер.  data::= structure  {  image\_block\_number: double-long-unsigned,  image\_block\_value: octet-string  }  После успешного вызова метода соответствующий бит атрибута image\_transferred\_blocks\_status имеет значение 1, а атрибут image\_first\_not\_transferred\_block\_number обновляется. |
| image\_verify | Проверяет целостность образа перед активацией.  data:: = integer (0)  Результатом вызова этого метода может быть success (успех), temporary\_failure (временный сбой) или other\_reason (другая причина). В случае неуспеха, результат проверки можно узнать, получив значение атрибута image\_transfer\_status.  В случае успеха, атрибут image\_to\_activate\_info содержит информацию об образе для активации.  Примечание 5: *коды Action-Result/Data-Access-Result xDLMS-сервисов указаны в IEC 62056-5-3:2016, пункт 8.* |
| image\_activate | Активирует образ.  data::= integer (0)  Если переданный образ ранее не проверялся, то это делается в рамках активации образа.  Результатом вызова этого метода может быть success (успех), temporary\_failure (временный сбой) или other\_reason (другая причина). В случае неуспеха, результат проверки можно узнать, получив значение атрибута image\_transfer\_status.  Примечание 6: *коды Action-Result/Data-Access-Result xDLMS-сервисов указаны в IEC 62056-5-3:2016, пункт 8.* |

## **COSEM Interface Class: Activity calendar (class\_id:20)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity calendar (Class\_id=20)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. calendar\_name\_active 3. season\_profile\_active 4. week\_profile\_table\_active 5. day\_profile\_table\_active 6. calendar\_name\_passive 7. season\_profile\_passive 8. week\_profile\_table\_passive 9. day\_profile\_table\_passive 10. activate\_passive\_calendar\_time | Static  Static  Static  Static  Static  Static  Static  Static  Static  Static | octet-string  octet-string  array  array  array  octet-string  array  array  array  octet-string | X  X+0x08  X+0x10  X+0x18  X+0x20  X+0x28  X+0x30  X+0x38  X+0x40  X+0x48 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. activate\_passive\_calendar (data) | | Необязательный | X+0x50 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Activity calendar» |
| calendar\_name\_active | Обычно содержит идентификатор, описывающий набор сценариев, активированных объектом. |
| season\_profile\_active | Содержит список сезонов, определенных датой их начала и конкретным week\_profile, который будет выполнен. Список сортируется по season\_start (в порядке возрастания);  array season  season::= structure  {  season\_profile\_name: octet-string,  season\_start: octet-string,  week\_name: octet-string  }  Где:  – season\_profile\_name – определяемое пользователем имя, идентифицирующее текущий season\_profile;  – season\_start определяет время начала сезона, отформатированное как указанно в 4.6.1 для даты-времени. В season\_start разрешены подстановочные знаки (маски, шаблоны). Если все поля являются подстановочными знаками, сезон никогда не начнется. Когда используются подстановочные знаки, необходимо проявлять особую осторожность, чтобы избежать конфликтов параметризации, т.е. один и тот же season\_start для двух различных сезонов;  *Обратите внимание, что текущий сезон прекращается по season\_start следующего сезона.*  – week\_name определяет активный в этом сезоне week\_profile. |
| week\_profile\_table\_active | Содержит массив week\_profiles для использования в разных сезонах. Для каждого файла week\_profile определяется файл day\_profile для каждого дня недели.  array week\_profile  week\_profile::= structure  {  week\_profile\_name: octet-string,  monday: day\_id,  tuesday: day\_id,  wednesday: day\_id,  thursday: day\_id,  friday: day\_id,  saturday: day\_id,  sunday: day\_id  }  day\_id: unsigned  Где:  – week\_profile\_name – определяемое пользователем имя, идентифицирующее текущий week\_profile;  – monday определяет day\_profile, действительный в понедельник;  – …  – sunday определяет day\_profile, действительный в воскресенье. |
| day\_profile\_table\_active | Содержит массив day\_profiles, идентифицируемый по day\_id. Для каждого файла day\_profile задается список запланированных действий, определяемый скриптом для выполнения и соответствующим временем активации (start\_time). Список сортируется по start\_time.  array day\_profile  day\_profile::= structure  {  day\_id: unsigned,  day\_schedule: array day\_profile\_action  }  day\_profile\_action::= structure  {  start\_time: octet-string,  script\_logical\_name: octet-string,  script\_selector: long-unsigned  }  Где:  – day\_id – это определяемый пользователем идентификатор, определяющий текущий day\_profile;  - start\_time: определяет время выполнения скрипта (без подстановочных знаков); формат соответствует правилам, установленным в 4.6.1 для time;  – script\_logical\_name: определяет logical\_name объекта «Script table»;  – script\_selector: определяет script\_identifier исполняемого скрипта (который будет выполнен). |
| calendar\_name\_passive | Содержит идентификатор, описывающий набор сценариев, неактивированных объектом. |
| season\_profile\_passive | Содержит список сезонов, определенных датой их начала и конкретным week\_profile, который будет выполнен после активации. Список сортируется по season\_start (в порядке возрастания).  См. season\_profile\_active. |
| week\_profile\_table\_passive | Содержит массив week\_profiles для использования в разных сезонах. Для каждого файла week\_profile определяется файл day\_profile для каждого дня недели.  См. week\_profile\_table\_active. |
| day\_profile\_table\_passive | Содержит массив day\_profiles, идентифицируемый по day\_id.  См. day\_profile\_table\_active. |
| activate\_passive\_calendar\_time | Определяет время, когда сам объект вызывает определенный метод activate\_passive\_calendar. Определение "not specified" (не указано) во всех полях атрибута деактивируют этот автоматизм. Частичное использование "not specified" только в некоторых полях даты и времени не допускается.  octet-string, отформатированная, как указано в 4.6.1 для date-time (даты-времени). |
| **Методы** | |
| activate\_passive\_calendar (data) | Этот метод копирует все атрибуты с именем ...\_passive в соответствующие атрибуты с именем ...\_active.  data:: = integer (0) |

## **COSEM Interface Class: Security setup (class\_id:64)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Security setup (Class\_id=64)** | | **Версия=0** | **Короткое имя** |
| **Атрибуты** | | **Тип данных** |
| 1. logical\_name 2. security\_policy 3. security\_suite 4. client\_system\_title 5. server\_system\_title | Static  Static  Static  Dynamic  Static | octet-string  enum  enum  octet-string  octet-string | X  X+0x08  X+0x10  X+0x18  X+0x20 |
| **Определенные методы** | | **Обязательный/Необязательный** |  |
| 1. security\_activate 2. global\_key\_transfer | | Необязательный  Необязательный | X+0x28  X+0x30 |
| **Занятый диапазон коротких имен (адресное пространство):** | | |  |

|  |  |
| --- | --- |
| **Атрибуты** | |
| logical\_name | Идентифицирует экземпляры объектов IC «Security setup». |
| security\_policy | Задает применение алгоритма аутентификации и/или шифрования, поставляемого с параметром security\_suite.  enum:  (0) Ничего,  (1) Все сообщения должны быть заверены (аутентифицированы),  (2) Все сообщения должны быть зашифрованы,  (3) Все сообщения должны проходить проверку подлинности и шифроваться,  (4) ... (15) зарезервированы. |
| security\_suite | Указывает алгоритм аутентификации, шифрования и передачи ключей.  enum:  (0) AES-GCM-128 для проверки подлинности и шифрования и AES-128 для упаковки ключей,  (1) ... (15) зарезервированы. |
| client\_system\_title | Несет (текущий) заголовок (название) системы клиента:  – в среде S-FSK PLC инициатор активности отправляет заголовок своей системы по протоколу CIASE;  Примечание 1: *вместе с этим передается атрибут active\_initiator объекта инициатора S-FSK активности; см. 5.8.4;*  – в ходе установления подтвержденной или неподтвержденной АА, он (заголовок системы клиента) передается в поле calling-AP-title AARQ APDU;  – если заголовок (название) системы клиента уже отправлены в течение процесса регистрации, как в случае S-FSK PLC профиля, заголовок системы клиента, передаваемый в AARQ APDU должен быть таким же. В противном случае АА отклоняется и отправляется соответствующая диагностическая информация.  – в предварительно установленной АА, он (заголовок системы клиента) может быть записан (передан) клиентом, используя незащищенный SET/Write сервис. |
| server\_system\_title | Несет название (заголовок) системы сервера.  – в среде S-FSK PLC сервер отправляет название своей системы во время процесса обнаружения (discover), используя протокол CIASE;  – в ходе установления подтвержденной АА, он (заголовок системы сервера) передается в поле responding-AP-title AARE APDU.  Этот атрибут должен быть только для чтения. |
| **Методы** | |
| security\_activate | Активирует и укрепляет (усиляет) политику безопасности:  enum:  (0) Ничего,  (1) Все сообщения должны проходить проверку подлинности,  (2) Все сообщения должны быть зашифрованы,  (3) Все сообщения должны проходить проверку подлинности и шифроваться.  Новая политика безопасности применяется, как только вызов метода был успешно подтвержден.  Примечание 2: *политика безопасности может быть только усилена.* |
| global\_key\_transfer | Обновляет один или несколько глобальных ключей. Параметр data включает в себя зашифрованный ключ данных. Ключевые данные включают идентификаторы ключей и сами ключи.  array key\_data  key\_data::= structure  {  key\_id: enum:  (0) глобальный одноадресный ключ шифрования,  (1) глобальный ключ шифрования передачи,  (2) ключ проверки подлинности  key\_wrapped: octet-string  }  Алгоритм обертывания ключей определяется пакетом безопасности. KEK – это главный ключ (мастер ключ).  Новые ключи действительны (вступают в силу), как только вызов метода был успешно подтвержден. |

# **Общие типы данных**

Ниже приводится список типов данных, используемых для атрибутов объектов COSEM.

|  |  |  |  |
| --- | --- | --- | --- |
| **Обозначение типа** | **Тэгa** | **Описание** | **Диапазон значений** |
| **-- simple data types** | | | |
| null-data | [0] | Пустое значение. |  |
| boolean | [3] | Булевый (логический, двоичный). | TRUE/FALSE |
| bit-string | [4] | Упорядоченная последовательность двоичных значений. |  |
| double-long | [5] | 32-х разрядное целое (Integer32) | -2 147 483 648 …  2 147 483 647 |
| double-long-unsigned | [6] | 32-х разрядное целое, без знака (Unsigned32) | 0 … 4 294 967 295 |
|  | [7] | Тэг «floating-point» тип в IEC 61334-4-41:1996, не используется в DLMS/COSEM. См. тэг [23] и [24] |  |
| octet-string | [9] | Упорядоченная последовательность октетов (8-ми битных байтов) |  |
| visible-string | [10] | Упорядоченная последовательность ASCII символов |  |
|  | [11] | Тэг «time» тип в IEC 61334-4-41:1996, не используется в DLMS/COSEM. См. тег [27] |  |
| utf8-string | [12] | Упорядоченная последовательность символов в кодировке UTF-8 |  |
| bcd | [13] | Десятичное в двоичном коде |  |
| integer | [15] | 8-ми разрядное целое (Integer8) | -128 … 127 |
| long | [16] | 16-ти разрядное целое (Integer16) | -32 768 … 32 767 |
| unsigned | [17] | 8-ми разрядное целое, без знака (Unsigned8) | 0 … 255 |
| long-unsigned | [18] | 16-ти разрядное целое, без знака (Unsigned16) | 0 … 65 535 |
| long64 | [20] | 64-х разрядное целое (Integer64) | - 263 … 263-1 |
| long64-unsigned | [21] | 64-х разрядное целое, без знака (Unsigned64) | 0 … 264-1 |
| enum | [22] | Элементы типа перечисление определяются в описании атрибута или описании метода в разделах спецификации интерфейсного класса COSEM. | 0 … 255 |
| float32 | [23] | OCTET STRING (SIZE(4)) | См. 4.1.6.2. |
| float64 | [24] | OCTET STRING (SIZE(8)) |
| date-time | [25] | OCTET STRING SIZE(12)) | См. 4.1.6.1. |
| date | [26] | OCTET STRING (SIZE(5)) |
| time | [27] | OCTET STRING (SIZE(4)) |
| **-- complex data types** | | | |
| array | [1] | Элементы массива определяются в описании атрибута или метода в разделах спецификации интерфейсного класса COSEM. |  |
| structure | [2] | Элементы структуры определяются в описании атрибута или метода в разделах спецификации интерфейсного класса COSEM. |  |
| compact array | [19] | Предоставляет альтернативу, компактное кодирование сложных (комплексных) данных. |  |
| -- CHOICE |  | Для некоторых атрибутов COSEM интерфейсных объектов, тип данных может быть выбран при создании экземпляра, на этапе реализации сервером COSEM. Сервер всегда возвращают тип данных и значение каждого атрибута, таким образом вместе с логическим именем обеспечивается однозначное его толкование. Список возможных типов данных, определяется в разделе «Описание атрибута» спецификации интерфейсного класса COSEM. | |
| a Тэги приводятся, как определено в пункте 9.5 (DLMS UA 1000-2 Ed. 8.0:2014). | | | |

# **Scalar\_unit**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Enumerated values** | | **Unit** | **Quantity** | | **Unit name** | **SI definition (comment)** |
| (1) | a | | | time | year | |
| (2) | mo | | | time | month | |
| (3) | | wk | time | | week | 7\*24\*60\*60 s |
| (4) | | d | time | | day | 24\*60\*60 s |
| (5) | | h | time | | hour | 60\*60 s |
| (6) | | min. | time | | min | 60 s |
| (7) | | s | time (t) | | second | s |
| (8) | | ° | (phase) angle | | degree | rad\*180/ |
| (9) | | °C | temperature (T) | | degree-celsius | K-273.15 |
| (10) | currency | | | (local) currency | | |
| (11) | | m | length (l) | | metre | m |
| (12) | | m/s | speed (v) | | metre per second | m/s |
| (13) | | m3 | volume (V)  rV , meter constant or pulse value (volume) | | cubic metre | m3 |
| (14) | | m3 | corrected volume | | cubic metre | m3 |
| (15) | | m3/h | volume flux | | cubic metre per hour | m3/(60\*60s) |
| (16) | | m3/h | corrected volume flux | | cubic metre per hour | m3/(60\*60s) |
| (17) | m3/d | | | volume flux | m3/(24\*60\*60s) | |
| (18) | m3/d | | | corrected volume flux | m3/(24\*60\*60s) | |
| (19) | | l | volume | | litre | 10-3 m3 |
| (20) | kg | | | mass (m) | kilogram | |
| (21) | N | | | force (F) | newton | |
| (22) | | Nm | energy | | newton meter | J = Nm = Ws |
| (23) | | Pa | pressure (p) | | pascal | N/m2 |
| (24) | | bar | pressure (p) | | bar | 105 N/m2 |
| (25) | | J | energy | | joule | J = Nm = Ws |
| (26) | | J/h | thermal power | | joule per hour | J/(60\*60s) |
| (27) | | W | active power (P) | | watt | W = J/s |
| (28) | VA | | | apparent power (S) | volt-ampere | |
| (29) | var | | | reactive power (Q) | var | |
| (30) | | Wh | active energy rW , active energy meter constant or pulse value | | watt-hour | W\*(60\*60s) |
| (31) | | VAh | apparent energy rS , apparent energy meter constant or pulse value | | volt-ampere-hour | VA\*(60\*60s) |
| (32) | | varh | reactive energy rB , reactive energy meter constant or pulse value | | var-hour | var\*(60\*60s) |
| (33) | | A | current (I) | | ampere | A |
| (34) | | C | electrical charge (Q) | | coulomb | C = As |
| (35) | | V | voltage (U) | | volt | V |
| (36) | | V/m | electric field strength (E) | | volt per metre | V/m |
| (37) | | F | | capacitance (C) | farad | C/V = As/V | |
| (38) | | Ω | | resistance (R) | ohm | Ω = V/A | |
| (39) | Ωm2/m | | | resistivity (ρ) | | Ω m | |
| (40) | | Wb | | magnetic flux (Φ) | weber | Wb = Vs | |
| (41) | | T | | magnetic flux density (B) | tesla | Wb/m2 | |
| (42) | | A/m | | magnetic field strength (H) | ampere per metre | A/m | |
| (43) | | H | | inductance (L) | henry | H = Wb/A | |
| (44) | | Hz | | frequency (*f, ω)* | hertz | 1/s | |
| (45) | 1/(Wh) | | | RW , active energy meter constant or pulse value | | | |
| (46) | 1/(varh) | | | RB , reactive energy meter constant or pulse value | | | |
| (47) | 1/(VAh) | | | RS , apparent energy meter constant or pulse value | | | |
| (48) | | V2h | | volt-squared hour, rU2h , volt-squared hour meter constant or pulse value | volt-squared-hours | V2(60\*60s) | |
| (49) | | A2h | | ampere-squared hour, rI2h , ampere-squared hour meter constant or pulse value | ampere-squared-hours | A2(60\*60s) | |
| (50) | | kg/s | | mass flux | kilogram per second | kg/s | |
| (51) | | S, mho | | conductance | siemens | 1/Ω | |
| (52) | K | | | temperature (T) | kelvin | | |
| (53) | 1/(V2h) | | | RU2h , volt-squared hour meter constant or pulse value | | | |
| (54) | 1/(A2h) | | | RI2h , ampere-squared hour meter constant or pulse value | | | |
| (55) | 1/m3 | | | RV , meter constant or pulse value (volume) | | | |
| (56) | percentage | | | % | | | |
| (57) | Ah | | | ampere-hours | Ampere-hour | | |
| ... | | | | | | | |
| (60) | Wh/m3 | | | energy per volume | 3,6\*103 J/m3 | | |
| (61) | J/m3 | | | calorific value, wobbe | | | |
| (62) | | Mol % | | molar fraction of gas composition | mole percent | (Basic gas composition unit) | |
| (63) | g/m3 | | | mass density, quantity of material | (Gas analysis, accompanying elements) | | |
| (64) | | Pa s | | dynamic viscosity | pascal second | (Characteristic of gas stream) | |
| (65) | | J/kg | | Specific energy  NOTE The amount of energy per unit of mass of a substance | Joule / kilogram | m2 . kg . s -2 / kg = m2 . s –2 | |
| …. | | | | | | | |
| (70) | dBm | | | Signal strength, dB milliwatt (e.g. of GSM radio systems) | | | |
| (71) | dbμV | | | Signal strength, dB microvolt | | | |
| (72) | dB | | | Logarithmic unit that expresses the ratio between two values of a physical quantity | | | |
| … | | | | | | | |
| (253) | | | | reserved | | | |
| (254) | other | | | other unit | | | |
| (255) | count | | | no unit, unitless, count | | | |

# **Система идентификации объектов (OBIS)**

Система идентификации объектов (**Object identification system – OBIS**) определяет идентификационные коды для широко используемых элементов данных в измерительных устройствах. OBIS предоставляет уникальные идентификаторы для всех данных внутри измерительного устройства, включая идентификаторы не только измеряемых значений, но и также идентификаторы абстрактных значений, используемые для конфигурации или получения информация о поведении измерительного оборудования.

Применительно к экземплярам интерфейсных классов COSEM, OBIS код используется в качестве логического имени объекта COSEM и однозначно идентифицирует информацию, которую данный объект представляет.

## **Структура кода OBIS**

OBIS код, являющийся значением первого атрибута любого объекта COSEM, представляет собой числовую комбинацию из шести групп цифр (А B C D E F). Каждая группа содержит число из диапазона от 0 до 255. Назначение каждой группы чисел приводится ниже.

### **Группа A**

Группа А (принимает значения от 0 до 15). Значение этой группы определяет тип энергоресурса, с которым связано измерение (электричество, тепло, газ, вода и т.п.) или абстрактные данные. Под абстрактными данными понимается та информация, которая присуща прибору учета, а не измеряемой величине. Например, к абстрактным объектам относятся объекты, представляющие имя логического устройства, объекты ассоциации, объект, представляющий версию прошивки прибора учета и др.

|  |  |
| --- | --- |
| **Значения группы A** | |
| 0 | Абстрактные объекты. |
| 1 | Объекты, связанные с электричеством. |
| … |  |
| 4 | Объекты, связанные с учетом распределения тепла. |
| 5, 6 | Объекты, связанные с тепловой энергией. |
| 7 | Объекты, связанные с газом. |
| 8 | Объекты, связанные с холодной водой. |
| 9 | Объекты, связанные с горячей водой. |
| … |  |
| 15 | Другие ресурсы. |
| All other | Зарезервировано. |

### **Группа B**

Группа В (принимает значения от 0 до 255). Значение этой группы, как правило, определяет номер измерительного канала, но также может определять и номер коммуникационного канала, а в некоторых случаях, и другие элементы. Интерпретация значений этой группы не зависит от значения группы A.

|  |  |
| --- | --- |
| **Значения группы B** | |
| 0 | Канал не указан |
| 1…64 | Канал 1..64 |
| 65…127 | Коды определяются программой. |
| 128…199 | Коды определяются производителем. |
| 200…255 | Зарезервировано. |

### **Группа С**

Группа C (принимает значения от 0 до 255). Значение этой группы определяет наименование измеряемой величины (ток, напряжение, мощность, объем, температура) или абстрактные данные. Интерпретация значения этой группы зависит от значения группы A. Методы обработки, классификации и хранения определяются значениями групп D, E и F. Для абстрактных данных значения групп D, E и F обеспечивают дальнейшую классификацию данных определенных значениями групп A, B и C.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Значения группы C – абстрактные объекты (A = 0)** | | | | | |
| 0…89 | | | Идентификаторы определяются контекстомa. | | |
| 93 | | | Идентификаторы определяются консорциумом. | | |
| 94 | | | Идентификаторы определяются страной. | | |
| 96 | | | Общие и обслуживающие объекты. | | |
| 97 | | | Объекты регистра ошибок – абстракция. | | |
| 98 | | | Список объектов – абстракция. | | |
| 99 | | | Объекты профилей данных – абстракция. | | |
| … | | |  | | |
| 127 | | | Неактивные объектыb. | | |
| 128…199, 240 | | | Коды определяются производителем. | | |
| All other | | | Зарезервировано. | | |
| a | | | Идентификаторы, определяемые контекстом, идентифицируют объекты характерные для определенного протокола и/или приложения. | | |
| b | | | Неактивный объект – это объект, который определен и присутствует в приборе учета, но не обладает функциональностью (не определена функциональность). | | |
| **Значения группы C – объекты, связанные с электричеством (A = 1)** | | | | | |
| 0 | | | Объекты общего назначения. | | |
| ΣLi | L1 | L2 | | L3 | (See also Note 2) |
| 1 | 21 | 41 | | 61 | Активная энергия + (QI+QIV) |
| 2 | 22 | 42 | | 62 | Активная энергия – (QII+QIII) |
| 3 | 23 | 43 | | 63 | Реактивная энергия + (QI+QII) |
| 4 | 24 | 44 | | 64 | Реактивная энергия – (QIII+QIV) |
| 5 | 25 | 45 | | 65 | Реактивная энергия QI |
| 6 | 26 | 46 | | 66 | Реактивная энергия QII |
| 7 | 27 | 47 | | 67 | Реактивная энергия QIII |
| 8 | 28 | 48 | | 68 | Реактивная энергия QIV |
| 9 | 29 | 49 | | 69 | Суммарная энергия + (QI+QIV) (See also Note 3) |
| 10 | 30 | 50 | | 70 | Суммарная энергия – (QII+QIII) |
| 11 | 31 | 51 | | 71 | Ток: любая фаза (C = 11) / Li фаза a (C= 31, 51, 71) |
| 12 | 32 | 52 | | 72 | Напряжение: любая фаза (C = 12) / Li фаза a (C= 32, 52, 72) |
| 13 | 33 | 53 | | 73 | Коэффициент мощности (See also Note 4) |
| 14 | 34 | 54 | | 74 | Частота сети |
| 15 | 35 | 55 | | 75 | Активная энергия (abs(QI+QIV)+(abs(QII+QIII)) a |
| 16 | 36 | 56 | | 76 | Активная энергия (abs(QI+QIV)-abs(QII+QIII)) |
| 17 | 37 | 58 | | 77 | Активная энергия QI |
| 18 | 38 | 58 | | 78 | Активная энергия QII |
| 19 | 39 | 59 | | 79 | Активная энергия QIII |
| 20 | 40 | 60 | | 80 | Активная энергия QIV |
| 81 | | | Аспект (точка зрения) b | | |
| 82 | | | Безразмерные показатели качества (скачки или спады) | | |
| 83 | | | Наводки и потери в линии c | | |
| 84 | | | ΣLi Коэффициент мощности – (See also Note 4) | | |
| 85 | | | L1 Коэффициент мощности – | | |
| 86 | | | L2 Коэффициент мощности – | | |
| 87 | | | L3 Коэффициент мощности – | | |
| 88 | | | ΣLi Среднеквадратичное значение тока (QI+QII+QIII+QIV) | | |
| 89 | | | ΣLi Среднеквадратичное значение напряжения (QI+QII+QIII+QIV) | | |
| 90 | | | ΣLi Ток (алгебраическая сумма – без знака – значение токов во всех фазах). | | |
| 91 | | | L0 Ток (нейтраль) a | | |
| 92 | | | L0 Напряжение (нейтраль) a | | |
| 93 | | | Идентификаторы определяются консорциумом. | | |
| 94 | | | Идентификаторы определяются страной. | | |
| 96 | | | Общие и обслуживающие объекты – Электричество. | | |
| 97 | | | Объекты регистра ошибок – Электричество. | | |
| 98 | | | Список объектов – Электричество. | | |
| 99 | | | Объекты профиля данных – Электричество. | | |
| 100…127 | | | Зарезервировано. | | |
| 128...199, 240 | | | Коды определяются производителем. | | |
| All other | | | Зарезервировано. | | |
| Примечание 1 | | | Li количество – значение (будет измеряться) системы измерения подключенной между фазой i и опорной точкой. В 3-фазных 4-проводных системах, опорной точкой является нейтраль. В 3-фазных 3-проводных системах, опорной точкой является фаза L2. | | |
| Примечание 2 | | | ΣLi количество – суммарное значение замеренное во всей системе. | | |
| Примечание 3 | | | Если считается значение только по одной наблюдаемой энергии по четырем квадрантам, должно использоваться C = 9. | | |
| Примечание 4 | | | Величины коэффициента мощности при C = 13, 33, 53, 73 рассчитываются, как PF = Active power+ (C = 1, 21, 41, 61) / Apparent power+ (C = 9, 29, 49, 69) or PF = Active power– (C = 2, 22, 42, 62) / Apparent power- (C = 10, 30, 50, 70).  В первом случае знак положительный (нет знака), это значит, что коэффициент мощности направлен вовнутрь (получает) (PF+).  Во втором случае знак отрицательный, это значит, что коэффициент мощности направлен вовне (отдает) (PF–).  Величины коэффициента мощности C = 84, 85, 86 and 87 всегда рассчитываются, как PF– = Active power– / Apparent power–. Эта величина – коэффициент мощности направленный вовне, он не имеет знака. | | |
| a  Подробные сведения о расширенных кодах, см. BB 7.5.3.3.  b  Подробные сведения о расширенных кодах, см. BB 7.5.3.4.  c  Подробные сведения о расширенных кодах, см. BB 7.5.3.5. | | | | | |

### **Группа D**

Группа D (принимает значения от 0 до 255). Значение этой группы определяет типы, или результат обработки физических величин, определенных значениями групп А и С, в соответствии с различными конкретными алгоритмами.

### **Группа E**

Группа E (принимает значения от 0 до 255). Значение этой группы определяет дальнейшую обработку или классификацию величин определенных значениями групп A – D.

### **Группа F**

Группа F (принимает значения от 0 до 255). Значение этой группы определяет накопленные данные, идентифицируемые через значения групп A – E, в соответствии с различными расчетными периодами.

# **Variable-Access-Specification**

The Variable-Access-Specification parameter for the ReadRequest and WriteRequest services can be modified by adding three new options:

· [5] block-number-access

Option used to access the next block when a block responds to a Read.Request.

· [6] data-block-access-read.

Option used to send a Read.Request with a sequence number requiring than 239 bytes.

· [7] data-block-access-write

Option used to send a Write.Request with a sequence number and a data number requiring more than 239 bytes.

The representation ASN.1 of the Variable-Access-Specification variable is as follows:

Variable-Access-Specification::= CHOICE

{

variable-name [2] IMPLICIT ObjectName

parameterized-access [4] IMPLICIT SEQUENCE

{

variable-name ObjectName,

selector Unsigned8

parameter Data

}

block-number-access [5] IMPLICIT SEQUENCE

{

block-number Unsigned16

}

data-block-access-read [6] IMPLICIT SEQUENCE

{

last-block BOOLEAN

block-number Unsigned16,

raw-data OCTET STRING

}

data-block-access-write [7] IMPLICIT SEQUENCE

{

last-block BOOLEAN

block-number Unsigned16,

}

}

The ReadResponse and WriteResponse service primitives are modified:

ReadResponse ::= SEQUENCE OF CHOICE

{

data [0] Data,

data-access-error [1] IMPLICIT Data-Access-Result

data-block-result [2] SEQUENCE

{

last-block BOOLEAN

block-number Unsigned16

raw-data OCTET STRING

}

block-number [3] Unsigned 16

}

WriteResponse ::= SEQUENCE OF CHOICE

{

success [0] IMPLICIT NULL,

data-access-error [1] IMPLICIT Data-Access-Result

block-number [2] Unsigned16

}