# **HDLC Definition**

**<Разделитель кадров>** (Frame Delimiter)

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

**<Формат кадра>** (Frame Format)

***FF – (2 байта)****, [****15..12****] биты –* **<Тип кадра>** *(FrameType), всегда "1010" (0xA). Возможные типы кадров: (I frame) READ REQUEST, (I frame) READ REASPONSE, (RR frame) RECEIVE READY, [****11****] –* **<Признак сегментирования кадра>** *(Segmentation). Возможные значения поля: 0 – единственный (последний) кадр, 1 – есть следующий кадр, [10..8] – не используются (RFU), [****7..0****] –* **<Длина кадра в байтах>** *(Length). При расчете длины кадра флаги начала и конца кадра не учитываются.*

**<Адресное поле>** (AddressField)

***AF – (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) – (****1 байт****), значение идентифицирует клиента. Зарезервированные адреса клиента: 0x00 – Нет станции (No-station), 0x01 – Процесс управляется клиентом (Client Management Process), 0x10 – Общий доступ (Public Client – самый низкий уровень безопасности). Адрес клиента используется для выбора уровня доступа.* ***При кодировании адреса клиента резервируется младший значащий бит, который затем устанавливается в единицу.***

**<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 (****1 байт****). Есть только верхний адрес (logical device address). В некоторых случаях, нижняя часть может быть опущена.*

*Двухбайтовая адресация (****2 байта****). Есть верхний адрес (logical device address) – 1 байт и нижний адрес (physical device address) – 1 байт.*

*Четырехбайтовая адресация (****4 байта****). Есть верхний адрес – 2 байта и нижний адрес – два байта.*

*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.*

**<Управляющее поле>** (Control)

***CF*** *– (****1 байт****), задает тип команды или ответа, а также значения счетчиков, отправленных/принятых кадров, [****7..5****] биты –* **<значение счетчика принятых кадров>** *(RFNumber), [****4****] –* **<бит опроса/завершения>** *(PF – poll/final), [****3..1****] –* **<значение счетчика отправленных кадров>** *(SFNumber), [****0****] –* **<тип кадра>** *(FrameType). Возможные типы кадров и соответствующие им значения:*

*0 – I – Информационный кадр (Information frame).*

*1 –* ***RR*** *– P\F – Готов к приему (Receive ready – code:* ***odd1****)*

*1 –* ***RNR*** *– P\F – Не готов к приему (Receive not ready – code:* ***odd5****)*

*1 –* ***SNRM*** *– P – Установить режим нормального ответа (Set Normal Response Mode – code:* ***93****)*

*1 –* ***DISC*** *– P – Разъединить (Disconnect – code:* ***53****)*

*1 –* ***UA*** *– F – Ненумерованное подтверждение (Unnumbered Acknowledgment – code:* ***73****)*

**<Код целостности заголовка>** (Header Check Sequence)

***HCS*** *– (****2 байта****) (CRC-16/X-25).*

**<Информационное поле>** (Information)

***Максимальное значение длины*** *информационного поля* ***2030 байт*** *(значение* ***по умолчанию 128 байт****)*

**<Код целостности HDLC кадра>** (Frame Check Sequence)

***FCS*** *– (****2 байта****) (CRC-16/X-25).*

# **APDU Definition**

## **<Поле логического управления каналом>** (LLC bytes)

***LLC (Logical link control)*** *– (****3 байта****), все APDU кадры начинаюстя LLC полем:*

*для нисходящего (Client → Server) – 0x****E6E600****,*

*для восходящего (Server → Client) – 0x****E6E700****.*

## **<Тип PDU>** (PDU type)

***PDU type*** *– (****1 байт****), специфицирует Блок данных COSEM протокола прикладного уровня (COSEM APDU – COSEM Application Layer Protocol Data Unit):*

*Запрос на чтение (****Read Request****– APDU [****5****]) –**0x****05***

*Ответ на запрос чтения (****Read Response****– APDU [****12****]) – 0x****0C***

*Запрос на запись (****Write Request****– APDU [****6****]) – 0x****06***

*Ответ на запрос записи (****Write Response****– APDU [****13****]) – 0x****0D***

*Запрос ассоциации приложения (****Application Association Request – AARQ –*** *APDU* ***[96]****) – 0x****60***

*Ответ на запрос ассоциации приложения (****Application Association Response – AARE –*** *APDU* ***[97]****)* ***–*** *0x****61***

*Запрос получения данных (****Get-Request*** *– APDU [****192****]) – 0x****C0***

*Ответ на запрос получения данных* ***(Get-Response*** *– APDU [****196****] – 0x****C4****)*

*Если Запрос на чтение –**0x****05***

### **<Количество элементов в последовательности>** (Quantity)

***Qty*** *– (****1 байт****), специфицирует количество элементов указанных в APDU кадре (блоке).*

### **<Определение переменной доступа>** (VariableAccessSpecification)

***VAS*** *– (****1 байт****), специфицирует тип доступа (переменную доступа):*

*0x****02*** *– доступ по имени объекта (variable-name* ***ObjectName****)*

*0x****03*** *– детальный доступ (detailed-access) –* ***не используется*** *в DLMS/COSEM*

*0x****04*** *– параметризованный доступ (****Parameterized-Access****)*

*0x****05*** *– доступ по номеру блока (****Block-Number-Access****)*

*0x****06*** *– доступ на по-блочное чтение данных (****Read-Data-Block-Access****)*

*0x****07*** *– доступ на по-блочную запись данных (****Write-Data-Block-Access****)*

*Если VAS = 0x****02****:*

#### **<Короткое имя объекта>** (Object Short Name)

***ObjectShortName –*** *(****2 байта****), определяет короткое имя (Short Name) запрашиваемого объекта: базовое короткое имя объекта + смещение запрашиваемого атрибута (например, 0xFD08 = 0xFD00 (объект: Logical Device Name) + 0x08 (атрибут: value).*

*Если Запрос ассоциации приложения – 0x****60***

### **<Длина содержимого AARQ>** (AARQ length)

***AARQ length*** *– (****1 байт****), кодирование длины содержимого AARQ – 0x****1D****.*

### **<Имя контекста приложения>** (application-context-name), кодируется как приложение будет обращаться к объектам: по коротким или по логическим именам.

*(****tag****) – (****1 байт****), кодирование тега tag [****1****]: application-context-name – 0x****A1****.*

*(****length****) – (****1 байт****), кодирование длины описания имени контекста приложения – 0x****09****.*

*(****ACSE type****) – (****1 байт****), types used in the fields of the ACSE (Association Control Service Element) APDUs (encoding of the choice for application-context-name)– 0x****06*** *(тип <**OBJECT IDENTIFIER>).*

*(****OBJECT IDENTIFIER length****) – (****1 байт****), длина идентификатора объекта – – 0x****07****.*

*(****OBJECT IDENTIFIER****) – (указанные выше* ***7 байт****),*

*для SN referencing – 0x****60857405080102*** *(обращение по короткому имени)*

*для LN referencing – 0x****60857405080101*** *(обращение по логическому имени)*

### **<Информация пользователя>** (user-information)

*(****tag****) – (****1 байт****), кодирование тега tag [30]: user-information – 0x****BE****.*

*(****length****) – (****1 байт****), кодирование длины описания информации пользователя – 0x****10****.*

*(****ACSE type****) – (****1 байт****), encoding the choice for user-information – 0x****04*** *(тип <OCTET STRING>).*

*(****OCTET STRING length****) – (****1 байт****), длина идентификатора объекта – – 0x****0E****.*

*(****user-information****) – (14* ***байт****), 0x****01000000065F1F0400007E1F04B0*** *(xDLMS InitiateRequest APDU) (в лог-файле Вектор-300: 01000000065F1F04001C1B200000).*

*Если Ответ на запрос ассоциации приложения – 0x****61***

### **<Длина содержимого AARE>** (AARE length)

***AARE length*** *– (****1 байт****), кодирование длины содержимого AARE – 0x****29****.*

### **<Имя контекста приложения>** (application-context-name), кодируется как приложение будет обращаться к объектам: по коротким или по логическим именам.

*(****tag****) – (****1 байт****), кодирование тега tag [****1****]: application-context-name – 0x****A1****.*

*(****length****) – (****1 байт****), кодирование длины описания имени контекста приложения – 0x****09****.*

*(****ACSE type****) – (****1 байт****), types used in the fields of the ACSE (Association Control Service Element) APDUs (encoding of the choice for application-context-name)– 0x****06*** *(тип <OBJECT IDENTIFIER>).*

*(****OBJECT IDENTIFIER length****) – (****1 байт****), длина идентификатора объекта – – 0x****07****.*

*(****OBJECT IDENTIFIER****) – (указанные выше* ***7 байт****),*

*для SN referencing – 0x****60857405080102*** *(обращение по короткому имени)*

*для LN referencing – 0x****60857405080101*** *(обращение по логическому имени)*

### **<Результат ассоциации>** (association-result)

*(****tag****) – (****1 байт****), кодирование тега tag [****2****]: association-result – 0x****A2****.*

*(****length****) – (****1 байт****), кодирование длины поля «результат ассоциации» – 0x****03****.*

*(****ACSE type****) – (****1 байт****), encoding the choice for association-result – 0x****02*** *(тип <INTEGER>).*

*(****association-result field length****) – (****1 байт****), длина поля «результат ассоциации» – 0x****01****.*

*(****association-result value****) – (****1 байт****), значение «результат ассоциации» – 0x****00****.*

### **<Диагностика источника результатов>** (result-source-diagnostic)

*(****tag****) – (****1 байт****), кодирование тега tag [****3****]: result-source-diagnostic – 0x****A3****.*

*(****length****) – (****1 байт****), кодирование длины поля «диагностика источника результатов» – 0x****05****.*

*(****RSD type****) – (****1 байт****), encoding the choice for result-source-diagnostic [1] – 0x****A1*** *(acse-service-user).*

*(****tag field length****) – (****1 байт****), длина поля «диагностика источника результатов» – 0x****03****.*

*(****ACSE type****) – (****1 байт****), encoding the choice for result-source-diagnostic type – 0x****02*** *(тип <INTEGER>).*

*(****target field length****) – (****1 байт****), длина поля «диагностика источника результатов» – 0x****01****.*

*(****result-source-diagnostic value****) – (****1 байт****), значение «диагностика источника результатов»:*

*успешно (success, no security and LLS: 0x****00****, no diagnostics provided);*

*ошибка (failure 1: 0x****02****, application-context-name not supported);*

*ошибка (failure 2: 0x****01****, no-reason-given);*

*успешно (success, HLS security (5): 0x****0E****, authentication required).*

### **<Информация пользователя>** (user-information)

*(****tag****) – (****1 байт****), кодирование тега tag [30]: user-information – 0x****BE****.*

*(****length****) – (****1 байт****), кодирование длины описания информации пользователя – 0x****10****.*

*(****ACSE type****) – (****1 байт****), encoding the choice for user-information – 0x****04*** *(тип <OCTET STRING>).*

*(****OCTET STRING length****) – (****1 байт****), длина идентификатора объекта – – 0x****0E****.*

*(****user-information****) – (14* ***байт****),*

*0x****0800065F1F040000501F01F40007*** *(для LN referencing);*

*0x****0800065F1F04001C032001F4FA00*** *(xDLMS InitiateResponse для SN referencing) (в лог-файле Вектор-300: 0800065F1F04001C1320007AFA00).*

<тип/формат ответа> – (**1 байт**)

<*Тип ответа – Data-Block-Result*>

<наличие следующего блока> – признак последнего блока (**1 байт**):

*есть следующий блок – 0x****00*** (*last-block: false*)

*нет следующего блока – 0x****01*** (*last-block: true*)

<номер блока> – (**2 байта**)

<*Данные – Row Data*>

<длина данных в байтах> – (**1 байт**) (*перед ним может быть еще байт, например: 81, что это не понятно, возможно указатель на то, что длина будет записана одним байтом*).

<номер элемента в последовательности> – (**1 байт**)

<значение диагностики источника результата> – (**1 байт**) (*result-source-diagnostic value*)

<тип данных элемента данных> – (**1 байт**)

<длина элемента данных> – (**2 байта**) (*перед ним может быть еще байт, например: 82, возможно указатель на то, что длина будет записана двумя байтами*).

# **AssociationSN readout**

7EA0140002A0AD03BA7A50 *E6E600* **05 01 02 *FA 08*** 189E7E

7E - разделитель кадров (frame delimiter)

A014 - определение формата кадра (frame format) - несегментированный кадр - длина кадра: 20

0002A0AD - адрес сервера (server address) = 1:10326 - логический адрес сервера (upper part server address):физический адрес сервера (lower part setver address)

03 - адрес клиента (client address) = 1

BA - управляющее поле (control field)

7A50 - код целостности заголовка (header check sequence)

E6E600050102FA08 - информационное поле (information field – **APDU**):

*E6E600* – поле логического управления каналом (*LLC – Logical link control*) – входящий пакет.

**05** – тип PDU (*PDU type*) – Запрос на чтение (*Read Request*)

**01** – количество элементов в последовательности (*Quantity*)

**02** – определение переменной доступа (*VAS – VariableAccessSpecification*) – доступ по имени объекта (*variable-name ObjectName*)

**FA08** – короткое имя объекта (*Object Short Name*) – 0xFA00 *(logical\_name:* Association SN) + 0x08 (*object\_list*)

189E - код целостности кадра (frame check sequence)

7E - разделитель кадров (frame delimiter)

7EA082030002A0ADDA4B13 *E6E700* 0C 01 *02 00 0001* 6C *01 00* 01 *82* 014F 02 04 10 00 00 12 00 03 11 00 09 06 01 00 0F 08 00 FF 02 04 10 00 18 12 00 03 11 00 09 06 01 00 0F 08 01 FF 02 04 10 00 30 12 00 03 11 00 09 06 01 00 0F 08 02 FF 02 04 10 00 48 12 00 03 11 00 09 06 01 00 0F 08 03 FF 02 04 10 00 60 12 00 03 11 00 09 06 01 00 0F 08 04 FF 02 04 10 01 00 12 00 03 11 00 09 06 67987E

7E - разделитель кадров (frame delimiter)

A082 - определение формата кадра (frame format) - несегментированный кадр - длина кадра: -126

03 - адрес клиента (client address) = 1

0002A0AD - адрес сервера (server address) = 1:10326 - логический адрес сервера (upper part server address):физический адрес сервера (lower part setver address)

DA - управляющее поле (control field)

4B13 - код целостности заголовка (header check sequence)

E6E7000C01020000016C01000182014F02041000001200031100090601000F0800FF02041000181200031100090601000F0801FF02041000301200031100090601000F0802FF02041000481200031100090601000F0803FF02041000601200031100090601000F0804FF020410010012000311000906 - информационное поле (information field – **APDU**):

*E6E700* – поле логического управления каналом (*LLC – Logical link control*) – исходящий пакет.

**0C** – тип PDU (*PDU type*) – Ответ на запрос чтения (Read Response)

**01** – количество элементов в последовательности (*Quantity*)

**02** – тип/формат ответа (*ReadResponse [****2****] :* *Data-Block-Result*)

**00** – наличие следующего блока (*last-block – false, есть следующий блок*)

**0001** – номер блока (*block-number*)

**6C** – (возвращаемое описание объектов COSEM) – длина блока описания – ***108 байт***

**01** – первый элемент в последовательности (*item number*)

**00** – значение диагностики источника результата (*success – успешно*).

**01** – тип данных элемента данных (*data-types [****1****] : array*) – «массив»

***82*** **014F** – размерность массива (*количество элементов массива*) – 0x014F = **335** объектов (*82 перед значением, возможно указатель на то, что значение будет записано двумя байтами*).

**02** – тип данных элемента массива (объекта) (*data-types [****2****] : structure*) – «структура»

**04** – количество компонентов в структуре (*структура из четырех компонентов*).

**10** – тип первого в структуре компонента (*data-types [****16****] : integer 16*)

**0000** – значение первого в структуре компонента (base\_name – базовое короткое имя объекта) – 0x**0000**.

**12** – тип второго в структуре компонента (*data-types [****18****] : unsigned 16*)

**0003** – значение второго в структуре компонента (class\_id – идентификатор родительского класса) – 0x**0003** (*Register*).

**11** – тип третьего в структуре компонента (*data-types [****17****] : unsigned 8*)

**00** – значение третьего в структуре компонента (version – версия родительского класса) – 0x**00**.

**09** – тип четвертого в структуре компонента (*data-types [****9****] : octet-string*)

**06** – длина строки байтов (октетов) (octet string (*size(****6****)*) – размер 6 байт.

**01000F0800FF** – значение четвертого в структуре компонента (OBIS code – идентификационный код объекта) – 0x**01000F0800FF** (1.0.15.8.0.255 : Суммарная энергия |A|).

6798 - код целостности кадра (frame check sequence)

7E - разделитель кадров (frame delimiter)

7EA0140002A0AD03DC4A56 *E6E600* 05 01 05 00 01 A40E7E

7E - разделитель кадров (frame delimiter)

A014 - определение формата кадра (frame format) - несегментированный кадр - длина кадра: 20

0002A0AD - адрес сервера (server address) = 1:10326 - логический адрес сервера (upper part server address):физический адрес сервера (lower part setver address)

03 - адрес клиента (client address) = 1

DC - управляющее поле (control field)

4A56 - код целостности заголовка (header check sequence)

E6E6000501050001 - информационное поле (information field):

*E6E600* – поле логического управления каналом (*LLC – Logical link control*) – входящий пакет.

**05** – тип PDU (*PDU type*) – Запрос на чтение (*Read Request*)

**01** – количество элементов в последовательности (*Quantity*)

**05** – определение переменной доступа (*VAS – VariableAccessSpecification*) – доступ по номеру блока (VariableAccessSpecification [5] – Block-Number-Access)

**0001** – номер полученного блока (*recived block number*)

A40E - код целостности кадра (frame check sequence)

7E - разделитель кадров (frame delimiter)

7EA082030002A0ADFC7F57 *E6E700* 0C 01 02 00 00 02 6C 01 00 01 08 00 FF 02 04 10 01 18 12 00 03 11 00 09 06 01 00 01 08 01 FF 02 04 10 01 30 12 00 03 11 00 09 06 01 00 01 08 02 FF 02 04 10 01 48 12 00 03 11 00 09 06 01 00 01 08 03 FF 02 04 10 01 60 12 00 03 11 00 09 06 01 00 01 08 04 FF 02 04 10 02 00 12 00 03 11 00 09 06 01 00 02 08 00 FF 02 04 10 02 18 12 00 03 11 00 09 06 AA2F7E

7E - разделитель кадров (frame delimiter)

A082 - определение формата кадра (frame format) - несегментированный кадр - длина кадра: -126

03 - адрес клиента (client address) = 1

0002A0AD - адрес сервера (server address) = 1:10326 - логический адрес сервера (upper part server address):физический адрес сервера (lower part setver address)

FC - управляющее поле (control field)

7F57 - код целостности заголовка (header check sequence)

E6E7000C01020000026C*0100010800FF*0204100118120003110009060100010801FF0204100130120003110009060100010802FF0204100148120003110009060100010803FF0204100160120003110009060100010804FF0204100200120003110009060100020800FF020410021812000311000906 - информационное поле (information field):

*E6E700* – поле логического управления каналом (*LLC – Logical link control*) – исходящий пакет.

**0C** – тип PDU (*PDU type*) – Ответ на запрос чтения (Read Response)

**01** – количество элементов в последовательности (*Quantity*)

**02** – тип/формат ответа (*ReadResponse [****2****] :* *Data-Block-Result*)

**00** – наличие следующего блока (*last-block – false, есть следующий блок*)

**0002** – номер блока (*block-number*) – №2

**6C** – (возвращаемое описание объектов COSEM) – длина блока описания – ***108 байт***

**0100010801FF** – *значение четвертого* в структуре *шестого* компонента (OBIS code – идентификационный код объекта) – 0x**0100010801FF** (1.0.1.8.0.255 : Суммарная энергия +A [kWh], текущее значение).

**02** – тип данных элемента массива (объекта) (*data-types [****2****] : structure*) – «структура»

**04** – количество компонентов в структуре (*структура из четырех компонентов*).

**10** – тип первого в структуре компонента (*data-types [****16****] : integer 16*)

**0118** – значение первого в структуре компонента (base\_name – базовое короткое имя объекта) – 0x**0118**.

**12** – тип второго в структуре компонента (*data-types [****18****] : unsigned 16*)

**0003** – значение второго в структуре компонента (class\_id – идентификатор родительского класса) – 0x**0003** (*Register*).

**11** – тип третьего в структуре компонента (*data-types [****17****] : unsigned 8*)

**00** – значение третьего в структуре компонента (version – версия родительского класса) – 0x**00**.

**09** – тип четвертого в структуре компонента (*data-types [****9****] : octet-string*)

**06** – длина строки байтов (октетов) (octet string (*size(****6****)*) – размер 6 байт.

**0100010801FF** – значение четвертого в структуре компонента (OBIS code – идентификационный код объекта) – 0x**0100010801FF** (1.0.1.8.1.255: Суммарная энергия +A [kWh] тарифа T=1, текущее значение).

AA2F - код целостности кадра (frame check sequence)

7E - разделитель кадров (frame delimiter)

# **AssociationLN readout**

7EA01A025721324F91 *E6E600* C0 01 C1 00 0F 00 00 28 00 00 FF 02 00 91537E

7E - разделитель кадров (frame delimiter)

A01A - определение формата кадра (frame format) - несегментированный кадр - длина кадра: 26

0257 - адрес сервера (server address) = 1:43 - логический адрес сервера (upper part server address):физический адрес сервера (lower part setver address)

21 - адрес клиента (client address) = 16

32 - управляющее поле (control field)

4F91 - код целостности заголовка (header check sequence)

E6E600C001C1000F0000280000FF0200 - информационное поле (information field):

*E6E600* – поле логического управления каналом (*LLC – Logical link control*) – входящий пакет.

**C0** – тип PDU (*PDU type*) – Запрос получения данных (*PDU [192] –* *Get-Request*)

**01** – Нормальный запрос получения данных (*Get-Request [1] – Get-Request-Normal*)

**C1** – Вызов по идентификатору и индексу (*Get-Request-Normal [1] – InvokeIdAndPriority*)

**000F** – (**2 байта**) Идентификатор родительского класса (*ClassId 0x000F = 15 – Association LN*)

**0000280000FF** – (**6 байт**) Идентификатор объекта (*InstanceId – OBIS : 0.0.40.0.0.255 – Current association*)

**02** – Идентификатор атрибута (*AttributeId [2] : object\_list – список объектов*)

**00** – Флаг завершения описания атрибута (*AttributeDescriptor*)

9153 - код целостности кадра (frame check sequence)

7E - разделитель кадров (frame delimiter)

7EA01921025752515B *E6E700* C4 02 C1 00 00 00 00 01 00 02 01 50 99F47E

7E - разделитель кадров (frame delimiter)

A019 - определение формата кадра (frame format) - несегментированный кадр - длина кадра: 25

21 - адрес клиента (client address) = 16

0257 - адрес сервера (server address) = 1:43 - логический адрес сервера (upper part server address):физический адрес сервера (lower part setver address)

52 - управляющее поле (control field)

515B - код целостности заголовка (header check sequence)

E6E700C402C1000000000100020150 - информационное поле (information field):

*E6E700* – поле логического управления каналом (*LLC – Logical link control*) – исходящий пакет.

**C4** – тип PDU (*PDU type*) – Ответ на запрос получения данных (*PDU [196] –* *Get-Response*)

**02** – Ответ на запрос получения данных с блоком данных (*Get-Response* [2] – Get-Response-With-Datablock)

**C1** – Вызов по идентификатору и индексу (*Get-Response-Normal [1] – InvokeIdAndPriority*)

**00** – Признак последнего блока (*LastBlock Value=false – не последний блок*)

**00000001** – (**4 байта**) Номер блока (*BlockNumber 0x00000001 = 1*)

**00** – (*возможно дескриптор какой-то?*)

**02** – длина данных (*data length – 0x02 = 2 байта*)

**0150** – данные (*RawData 0x0150 = 336*) (*что это не понятно*).

99F4 - код целостности кадра (frame check sequence)

7E - разделитель кадров (frame delimiter)

7EA01402572154C7F6 *E6E600* C0 02 C1 00 00 00 01 51BE7E

7E - разделитель кадров (frame delimiter)

A014 - определение формата кадра (frame format) - несегментированный кадр - длина кадра: 20

0257 - адрес сервера (server address) = 1:43 - логический адрес сервера (upper part server address):физический адрес сервера (lower part setver address)

21 - адрес клиента (client address) = 16

54 - управляющее поле (control field)

C7F6 - код целостности заголовка (header check sequence)

E6E600C002C100000001 - информационное поле (information field):

*E6E600* – поле логического управления каналом (*LLC – Logical link control*) – входящий пакет.

**C0** – тип PDU (*PDU type*) – Запрос получения данных (*PDU [192] –* *Get-Request*)

**02** – Запрос получения следующего блока даных (*Get-Request [2] –* *Get-Request-Next*)

**C1** – Вызов по идентификатору и индексу (*Get-Request-Next [1] – InvokeIdAndPriority*)

**00000001** – (**4 байта**) Номер полученного блока (*RecivedBlockNumber 0x00000001 = 1*)

51BE - код целостности кадра (frame check sequence)

7E - разделитель кадров (frame delimiter)

7EA02A2102577478D6 *E6E700* C4 02 C1 00 00 00 00 02 00 13 02 04 12 00 08 11 00 09 06 00 00 01 00 00 FF 02 02 01 09 CA737E

7E - разделитель кадров (frame delimiter)

A02A - определение формата кадра (frame format) - несегментированный кадр - длина кадра: 42

21 - адрес клиента (client address) = 16

0257 - адрес сервера (server address) = 1:43 - логический адрес сервера (upper part server address):физический адрес сервера (lower part setver address)

74 - управляющее поле (control field)

78D6 - код целостности заголовка (header check sequence)

E6E700C402C1000000000200130204120008110009060000010000FF02020109 - информационное поле (information field):

*E6E700* – поле логического управления каналом (*LLC – Logical link control*) – исходящий пакет.

**C4** – тип PDU (*PDU type*) – Ответ на запрос получения данных (*PDU [196] –* *Get-Response*)

**02** – Ответ на запрос получения данных с блоком данных (*Get-Response* [2] – Get-Response-With-Datablock)

**C1** – Вызов по идентификатору и индексу (*Get-Response-Normal [1] – InvokeIdAndPriority*)

**00** – Признак последнего блока (*LastBlock Value=false – не последний блок*)

**00000002** – (**4 байта**) Номер блока (*BlockNumber 0x00000002 = 2*)

**00** – (*возможно дескриптор какой-то?*)

**13** – длина данных (*data length – 0x13 = 19 байт*)

**02** – тип данных элемента массива (объекта) (*data-types [****2****] : structure*) – «структура»

**04** – количество компонентов в структуре (*структура из четырех компонентов*).

**12** – тип первого в структуре компонента (*data-types [****18****] : unsigned 16*)

**0008** – значение первого в структуре компонента (class\_id – идентификатор родительского класса) – 0x**0008** (*Clock*).

**11** – тип второго в структуре компонента (*data-types [****17****] : unsigned 8*)

**00** – значение второго в структуре компонента (version – версия родительского класса) – 0x**00**.

**09** – тип третьего в структуре компонента (*data-types [****9****] : octet-string*)

**06** – длина строки байтов (октетов) (octet string (*size(****6****)*) – размер 6 байт.

**0000010000FF** – значение третьего в структуре компонента (OBIS code – идентификационный код объекта) – 0x**0000010000FF** (0.0.1.0.0.255: Дата и время).

**02** – тип четвертого в структуре компонента (*data-types [****2****] : structure*)

**02** – количество компонентов в структуре (структура из двух компонентов: *права доступа к атрибутам и методам объекта*).

**01** – тип первого в структуре компонента (data-types [1] : array)

**09** – размерность массива (количество элементов массива) – 0x09 = 9 элементов.

CA73 - код целостности кадра (frame check sequence)

7E - разделитель кадров (frame delimiter)

Для простого запроса по короткому имени:

**private void** resetFrame() {  
 **this**.**frame** = **new** HdlcFrame();  
 **this**.**frame**.setItems(Arrays.*asList*(  
 **new** FrameDelimiter(**this**.**frame**),  
 **new** FrameFormatDefinition(**this**.**frame**),  
 **new** Address(**this**.**frame**),  
 **new** Address(**this**.**frame**),  
 **new** ControlField(**this**.**frame**),  
 **new** HeaderCheckSequence(**this**.**frame**),  
 *// new InformationField(this.frame),* **new** LLC(**this**.**frame**),  
 **new** ApduTypeField(**this**.**frame**),  
 *// new Quantity(this.frame), // для запроса на чтение и ответа на запрос  
 // new VAS(this.frame), // для запроса на чтение  
 // new ShortName(this.frame), // для запроса на чтение  
 // Заголовок блока данных* **new** ReadResponseFormat(**this**.**frame**), *// для ответа на запрос на чтение* **new** LastBlock(**this**.**frame**), *// для ответа на запрос на чтение* **new** BlockNumber(**this**.**frame**), *// для ответа на запрос на чтение* **new** DataBlockLength(**this**.**frame**), *// для ответа на запрос на чтение  
 // конец заголовка блока (длина блока данных рассматривается в составе заголовка)* **new** ItemNumber(**this**.**frame**), *// для ответа на запрос на чтение* **new** ResultDiagnosticFlag(**this**.**frame**), *// для ответа на запрос на чтение* **new** ItemType(**this**.**frame**), *// для ответа на запрос на чтение* **new** ItemLength(**this**.**frame**), *// для ответа на запрос на чтение* **new** ItemValue(**this**.**frame**), *// для ответа на запрос на чтение* **new** RawData(**this**.**frame**), *// не разобранные данные* **new** FrameCheckSequence(**this**.**frame**),  
 **new** FrameDelimiter(**this**.**frame**)  
 ));  
}

Для APPLICATION ASSOCIATION REQUEST (AARQ)

**private void** resetFrame() {  
 **this**.**frame** = **new** HdlcFrame();  
 **this**.**frame**.setItems(Arrays.*asList*(  
 **new** FrameDelimiter(**this**.**frame**),  
 **new** FrameFormatDefinition(**this**.**frame**),  
 **new** Address(**this**.**frame**),  
 **new** Address(**this**.**frame**),  
 **new** ControlField(**this**.**frame**),  
 **new** HeaderCheckSequence(**this**.**frame**),  
 *// new InformationField(this.frame),* **new** LLC(**this**.**frame**),  
 **new** ApduTypeField(**this**.**frame**),  
 **new** AarqLength(**this**.**frame**), *// для запроса ассоциации приложения  
 // Элемент Службы Управления Ассоциацией tag [1]* **new** AcseTag(**this**.**frame**), *// для ассоциации приложения* **new** AcseLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseDataType(**this**.**frame**), *// для ассоциации приложения* **new** AcseDataLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseValue(**this**.**frame**), *// для ассоциации приложения  
 // Конец элемента Службы Управления Ассоциацией  
 // Элемент Службы Управления Ассоциацией tag [6]* **new** AcseTag(**this**.**frame**), *// для ассоциации приложения* **new** AcseLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseDataType(**this**.**frame**), *// для ассоциации приложения* **new** AcseDataLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseValue(**this**.**frame**), *// для ассоциации приложения  
 // Конец элемента Службы Управления Ассоциацией  
 // Элемент Службы Управления Ассоциацией tag [10]* **new** AcseTag(**this**.**frame**), *// для ассоциации приложения* **new** AcseLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseDataType(**this**.**frame**), *//* **new** AcseDataLength(**this**.**frame**), *//  
 // Конец элемента Службы Управления Ассоциацией  
 // Элемент Службы Управления Ассоциацией tag [11]* **new** AcseTag(**this**.**frame**), *// для ассоциации приложения* **new** AcseLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseValue(**this**.**frame**), *// для ассоциации приложения  
 // Конец элемента Службы Управления Ассоциацией  
 // Элемент Службы Управления Ассоциацией tag [12]* **new** AcseTag(**this**.**frame**), *// для ассоциации приложения* **new** AcseLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseDataType(**this**.**frame**), *//* **new** AcseDataLength(**this**.**frame**), *//  
 // Конец элемента Службы Управления Ассоциацией  
 // Элемент Службы Управления Ассоциацией tag [30]* **new** AcseTag(**this**.**frame**), *// для ассоциации приложения* **new** AcseLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseDataType(**this**.**frame**), *// для ассоциации приложения* **new** AcseDataLength(**this**.**frame**), *// для ассоциации приложения* **new** AcseValue(**this**.**frame**), *// комплексное значение, надо разбирать дальше  
 // Конец элемента Службы Управления Ассоциацией* **new** RawData(**this**.**frame**), *// не разобранные данные* **new** FrameCheckSequence(**this**.**frame**),  
 **new** FrameDelimiter(**this**.**frame**)  
 ));  
}

# **ApplicationAssociationRequest frame:**

**APDU frame:**

* **LLC** (3 байта),
* **APDU type** (1 байт)**:** [APPLICATION 0] == [ 0x**60** ] = [ 96 ]
* AARQ content (dinamic frame): AARQ TagsArray:
* **AarqLength** (1 байт) - длина запроса ассоциации приложения в байтах (AARQ Length),
* application-context-name [**1**], [ 0x**A1** ] = [ 161 ] : ACSE Tag structure:
* **AcseTagName** (1 байт),
* **AcseLength** (1 байт),
* **AcseDataType** (1 байт),
* **AcseDataLength** (1 байт),
* **AcseValue** (~ байт, определяется AcseDataLength)
* calling-AP-title [**6**], [ 0x**A6** ] = [ 166 ]
* sender-acse-requirements [**10**], [ 0x**8A** ] = [ 138 ] (*требования службы управления ассоциацией к отправителю*)
* authentication-mechanism-name [**11**], [ 0x**8B** ] = [ 139 ]
* calling-authentication-value [**12**], [ 0x**AC** ] = [ 172 ]
* user-information [**30**], Association-information [ 0x**BE** ] = [ 190 ]

# **ApplicationAssociationResponse frame:**

**APDU frame:**

* **LLC** (3 байта),
* **APDU type** (1 байт)**:** [APPLICATION 0] == [ 0x**61** ] = [ 97 ]
* **AarqLength** (1 байт) - длина запроса ассоциации приложения в байтах (AARQ Length),
* AARE content (dinamic frame): AARE TagsArray:
* application-context-name [**1**], [ 0x**A1** ] = [ 161 ]
* association-result [**2**], [ 0x**A2** ] = [ 162 ]
* associate-source-diagnostic [**3**], [ 0x**A3** ] = [ 163 ]
* user-information [**30**], Association-information [ 0x**BE** ] = [ 190 ]

# **ReadRequest frame:**

**APDU frame:**

* **LLC** (3 байта),
* **APDU type** (1 байт): [ReadRequest] == [ 0x**05** ] = [ 5 ],
* APDU content (nested frame): ReadRequest:
* **Quantity** (1 байт) (количество элементов в запросе)
* ArrayOfReadRequestElements [Quantity]:
* ReadRequestElement [1] (dinamic frame – structure):
* **VAS** (1 байт),
* **BlockNumber** (2 байта) (номер блока данных, для VAS[5] – BlockNumberAccess),
* **ShortName** (2 байт)
* ReadRequestElement [Quantity] (dinamic frame – structure):
* **VAS**
* **ShortName**
* **RawData** (неразобранные данные – технологическое поле)

# **ReadResponse frame:**

**APDU frame:**

* **LLC** (3 байта),
* **APDU type** (1 байт): [ReadResponse] == [ 0x**0С** ] = [ 12 ],
* APDU content (nested frame): ReadResponse:
* **Quantity** (1 байт) (количество элементов в последовательности),
* **ReadResponseFormat** (1 байт) (формат/тип ответа на запрос, например: [**2**] – data-block-result),
* **LastBlock flag** (1 байт) (признак последнего блока),
* **BlockNumber** (2 байта) (номер блока данных),
* **DataBlockLength** (1 байт) (длина блока данных в байтах),
* **Quantity** (1 байт) (количество элементов в блоке данных),
* ArrayOfReadResponseElements [Quantity]:
* ReadResponseElement [1] (dinamic frame – structure):
* **ResultDiagnosticFlag** (1 байт) значение диагностики источника результата (0x00 - success),
* **ItemType** (1 байт) тип данных элемента в последовательности,
* **ItemLength** (1 байт) длина элемента данных (или поля длины данных ItemLengthExtend),
* **ItemLengthExtend** (~ байт, определяется ItemLength) длина элемента данных (для байтовых строк в байтах),
* **ItemValue** (~ байт, определяется ItemType и ItemLength) значение элемента данных.
* ReadResponseElement [Quantity] (dinamic frame – structure):
* **ResultDiagnosticFlag** (1 байт) значение диагностики источника результата (0x00 - success),
* **ItemType** (1 байт) тип данных элемента в последовательности – [1] массив; [2] структура,
* **ItemLength** (1 байт) количество элементов в массиве | структуре,
* **ItemValue** массив:
* **ItemType** (1 байт) тип данных элемента в массиве | структуре,
* **ItemLength** (1 байт) длина элемента данных в байтах (для байтовых строк),
* **ItemValue** (~ байт, определяется ItemType и ItemLength) значение элемента данных:
* **RawData** (неразобранные данные – технологическое поле)

# **GetRequest frame:**

**APDU frame:**

* **LLC** (3 байта),
* **APDU type** (1 байт): [GetRequest] == [ 0x**C0** ] = [ 192 ],
* APDU content (nested frame): GetRequest:
* **GetRequestType** (1 байт) (тип GetRequest GR[1] – GetRequestNormal) [ 0x**01** ],
* **GetRequestVas** (1 байт) (тип GetRequestNormal GRN[1] – InvokeIdAndPriority) [ 0x**C1** ] = [ 193 ],
* **LongBlockNumber** (4 байта) (номер блока данных, для GR[2] – GetRequestNext),
* ArrayOfGetRequestElements [~]:
* GetRequestElement [1] (dinamic frame – structure):
* **ParentClassId** (2 байта) (идентификатор родительского класса),
* **ObjectId** (6 байт) (идентификатор объекта (OBIS код)),
* **AttributeId** (1 байт) (идентификатор атрибута),
* **AttributeDescriptor** (1 байт) (флаг завершения описания атрибута) [ 0x**00** ],
* GetRequestElement [~] (dinamic frame – structure):
* **ParentClassId** (2 байта) (идентификатор родительского класса),
* **ObjectId** (6 байт) (идентификатор объекта (OBIS код)),
* **AttributeId** (1 байт) (идентификатор атрибута),
* **AttributeDescriptor** (1 байт) (флаг завершения описания атрибута) [ 0x**00** ],
* **RawData** (неразобранные данные – технологическое поле)

# **GetResponse frame:**

**APDU frame:**

* **LLC** (3 байта),
* **APDU type** (1 байт): [GetResponse] == [ 0x**С4** ] = [ 196 ],
* APDU content (nested frame): GetResponse:
* **GetResponseType** (1 байт) (тип GetResponse GRE[2] – GetResponseWithDatablock) [ 0x**02** ],
* **GetResponseVas** (1 байт) (тип GetResponseWithDatablock GRDB[1] – InvokeIdAndPriority) [ 0x**C1** ] = [ 193 ],
* **LastBlock flag** (1 байт) (признак последнего блока),
* **LongBlockNumber** (4 байта) (номер блока данных),
* **ResultDiagnosticFlag** (1 байт) значение диагностики источника результата (0x00 - success),
* **DataBlockLength** (1 байт) (длина блока данных в байтах),
* ArrayOfGetResponseElements [~]:
* ReadResponseElement [1] (dinamic frame – structure):
* **ItemType** (1 байт) тип данных элемента в последовательности,
* **ItemLength** (1 байт) длина элемента данных (или поля длины данных ItemLengthExtend),
* **ItemLengthExtend** (~ байт, определяется ItemLength) длина элемента данных (для байтовых строк в байтах),
* **ItemValue** (~ байт, определяется ItemType и ItemLength) значение элемента данных.
* ReadResponseElement [~] (dinamic frame – structure):
* **ItemType** (1 байт) тип данных элемента в последовательности,
* **ItemLength** (1 байт) длина элемента данных (или поля длины данных ItemLengthExtend),
* **ItemLengthExtend** (~ байт, определяется ItemLength) длина элемента данных (для байтовых строк в байтах),
* **ItemValue** (~ байт, определяется ItemType и ItemLength) значение элемента данных.
* **RawData** (неразобранные данные – технологическое поле)

# **WriteRequest frame:**

**APDU frame:**

* **LLC** (3 байта),
* **APDU type** (1 байт): [WriteRequest] == [ 0x**06** ] = [ 6 ],
* APDU content (nested frame): WriteRequest:
* **Quantity** (1 байт) (количество элементов в запросе),
* **VAS** (1 байт) – тип доступа (переменная доступа),
* **ShortName** (2 байт) – короткое имя записываемого объекта,
* **Quantity** (1 байт) (количество элементов в блоке данных)
* ArrayOfWriteRequestElements [Quantity]:
* WriteRequestElement [1] (dinamic frame – structure):
* **ItemType** (1 байт) тип данных элемента в последовательности,
* **ItemLength** (1 байт) длина элемента данных (или поля длины данных ItemLengthExtend),
* **ItemLengthExtend** (~ байт, определяется ItemLength) длина элемента данных (для байтовых строк в байтах),
* **ItemValue** (~ байт, определяется ItemType и ItemLength) значение элемента данных.
* **RawData** (неразобранные данные – технологическое поле)

<PDU>

<WriteRequest>

<ListOfVariableAccessSpecification Qty="1" >

<VariableName Value="61760" />

</ListOfVariableAccessSpecification>

<ListOfData Qty="1" >

<Structure Qty="2" >

<OctetString Value="4913F7795D2B8012" />

<UInt32 Value="240" />

</Structure>

</ListOfData>

</WriteRequest>

</PDU>

# **WriteResponse frame:**

**APDU frame:**

* **LLC** (3 байта),
* **APDU type** (1 байт): [WriteResponse] == [ 0x**0D** ] = [ 13 ],
* APDU content (nested frame): WriteResponse:
* **Quantity** (1 байт) (количество элементов в запросе),
* ArrayOfWriteResponseElements [Quantity]:
* WriteResponseElement [1] (dinamic frame – structure):
* **ResultDiagnosticFlag** (1 байт) значение диагностики источника результата (0x00 - success),
* **RawData** (неразобранные данные – технологическое поле)

<PDU>

<WriteResponse Qty="1" >

<Success />

</WriteResponse>

</PDU>