Изображение герба Республики Казахстан

**НАЦИОНАЛЬНЫЙ СТАНДАРТ РЕСПУБЛИКИ КАЗАХСТАН**

**Единая сеть телекоммуникаций Республики Казахстан**

**Интернет вещей (IоТ)**

**Протокол беспроводной передачи данных LoRaWAN**

**СТ РК \_\_-202\_**

Настоящий проект стандарта не подлежит применению до его утверждения

**Комитет технического регулирования и метрологии**

**Министерства торговли и интеграции Республики Казахстан**

**(Госстандарт)**

**Нур-Султан**

**Предисловие**

**1 РАЗРАБОТАН** Акционерным обществом «Информационно-аналитический центр нефти и газа»

**ВНЕСЕН** Техническим комитетом по стандартизации №116 «Телекоммуникации» (в сфере услуг по передаче данных, организации каналов связи)

**2 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ** Приказом Председателя Комитета технического регулирования и метрологии Министерства торговли и интеграции Республики Казахстан № \_\_ от « » \_\_\_\_ 202\_ года

**3** Настоящем стандарте реализованы нормы Закона Республики Казахстан «О связи» от 5 июля 2004 года № 567-II

**4 ВВЕДЕН ВПЕРВЫЕ**

**5 СРОК ПЕРВОЙ ПРОВЕРКИ 202\_ год**

**ПЕРИОДИЧНОСТЬ ПРОВЕРКИ 5 лет**

*Информация об изменениях к настоящему стандарту публикуется в ежегодно издаваемом информационном каталоге «Документы по стандартизации», а текст изменений и поправок - в периодически издаваемых информационных указателях «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в периодически издаваемом информационном указателе «Национальные стандарты»*

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Комитета технического регулирования и метрологии Министерства торговли и интеграции Республики Казахстан

**Содержание**

|  |  |
| --- | --- |
| Введение |  |
| 1 Область применения |  |
| 2 Нормативные ссылки |  |
| 3 Термины и определения |  |
| 4 Обозначения и сокращения |  |
| 5 Общие положения |  |
| 5.1 Формат и структура |  |
| 5.2 Классы устройств LoRaWan |  |
| 6 Конечные устройства Классы А |  |
| 6.1 Формат сообщений на физическом уровне |  |
| 6.2 Формат сообщений на МАС-уровне |  |
| 6.3 МАС-команды |  |
| 6.4 Активация конечного устройства |  |
| 6.5 Задержка повторных передач |  |
| 7 Конечные устройства класса С |  |
| 7.1 Режим связи конечного устройства |  |
| 7.2 МАС-команды |  |
| 8 Примеры реализации |  |
| 9 Региональные параметры KZ |  |
| Библиография |  |

**Введение**

Интернет вещей (IoT) - это растущая экосистема физических устройств и повседневных объектов, встроенных в датчики и программное обеспечение, которое подключает их к сети для сбора и обмена данными между собой, а также с другими устройствами и системами с доступом в Интернет.

LoRaWAN - это протокол MAC (управление доступом к среде) и системная архитектура для сети LoRa. Он обеспечивает глобальный стандарт коммуникационных технологий LoRa® и обеспечивает беспроблемное взаимодействие устройств и сетей без сложных требований к установке.

Разработанный для приложений Интернета вещей (IoT) LoRa отвечает жизненно важным техническим требованиям, таким как двунаправленная связь, сквозная безопасность, мобильность и услуги локализации. LoRa обычно работает в безлицензионных субгигагерцевых радиодиапазонах, что дает каждому свободу построения сети LoRaWAN. Технология LoRa обеспечивает очень длительную передачу при исключительно низком энергопотреблении.

Датчики могут располагаться в помещении, на открытом воздухе или под землей и связываться напрямую со шлюзом в пределах диапазона до 50 км на открытой местности и до 10 км в городской среде. Нет необходимости в слишком сложном анализе покрытия.

Низкая скорость передачи данных и асинхронная связь обеспечивают низкое потребление энергии.

Датчики предназначены для отправки небольших битов данных, когда это необходимо, будь то событие или расписание, а срок службы батареи может составлять до 10 лет.

Адаптивная скорость передачи данных и многоканальный мульти-модемный трансивер в шлюзе обеспечивают одновременную передачу сообщений по нескольким каналам; это обеспечивает сеть LoRaWAN очень высокой пропускной способностью и масштабируемостью.

В настоящем стандарте принята схема размещения сервера сети, сервера приложений и сервера присоединения к сети Join на одной физической машине. Схема с раздельным размещением указанных компонент выходит за рамки настоящего стандарта.

**НАЦИОНАЛЬНЫЙ СТАНДАРТ РЕСПУБЛИКИ КАЗАХСТАН**

|  |
| --- |
| **Единая сеть телекоммуникаций Республики Казахстан**  **Интернет вещей (IоТ)**  **Протокол беспроводной передачи данных LoRaWAN** |

**Дата введения 202\_.\_\_.\_\_**

**1 Область применения**

Настоящий стандарт устанавливает основу для скоординированной работы по разработке инфраструктуры для области «интернета вещей» (loT, Internet of Things).

Настоящий стандарт определяет протокол передачи данных LoRaWAN и региональные параметры.

**2 Нормативные ссылки**

Для применения настоящего стандарта необходим следующий ссылочный нормативный документ по стандартизации:

СТ РК 3105-2017 Сети последующих поколений. Структура и функциональные модели архитектуры. Термины и определения для интернета вещей.

Примечание - При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов и классификаторов по ежегодно издаваемому информационному указателю «Нормативные документы по стандартизации» по состоянию на текущий год и соответствующим ежемесячно издаваемым информационным указателям, опубликованным в текущем году. Если ссылочный документ заменен (изменен), то при пользовании настоящим Стандартом следует руководствоваться замененным (измененным) документом. Если ссылочный документ отменен без замены, то положение, в котором дана ссылка на него, применяется в части, не затрагивающей эту ссылку.

***Проект, редакция 1***

**3 Термины и определения**

В настоящем стандарте применяются следующие термины с соответствующими определениями:

**3.1** **Интернет вещей** (Internet jf things), IoT: Глобальная информационно-коммуникационная инфраструктура, обеспечивающая возможность предоставления передовых (усовершенствованных) услуг путем соединения друг с другом (физических и виртуальных) вещей на основе существующих и развивающихся функционально совместимых информационно-коммуникационных технологий.

**3.2 Конечное устройство** (end-device): Программно-аппаратное решение, являющееся узлом сети, который может обмениваться сообщениями по протоколу LoRaWAN.

**3.3 Многоадресная рассылка** (multicast): Режим передачи нисходящего сообщения нескольким конечным устройствам одновременно.

**3.4 Восходящее сообщение** (uplink): Сообщение, передаваемое от конечного устройства к серверу сети.

**3.5 Нисходящее сообщение** (downlink): Сообщение, передаваемое от сервера сети к конечному устройству.

**4 Обозначения и сокращения**

В настоящем стандарте применяются следующие сокращения:

**RFU (**(Reserved For Future)**:** Зарезервировано для будущих применений.

**IoT** (Internet of Things): Интернет вещей.

**5 Общие положения**

**5.1 Формат и структура**

В настоящем стандарте используются следующие форматы записей:

-МАС-команды записываются в формате LinkCheckReq;

-константы записываются в формате RECEIVE\_DELAY1.

В настоящем стандарте структура сообщений изложена с учетом:

-порядка следования байт и бит для всех полей - «от младшего к старшему» (little-endian);

-бита RFU (Reserved For Future), обозначающего поле для будущего использования. По умолчанию данный бит должен быть установлен на нуль передатчиком сообщения и должен быть проигнорирован на приемной стороне.

**5.2 Классы устройств LoRaWan**

Все конечные устройства должны реализовать базовую функциональность класса А, определенного в настоящем стандарте.

Некоторые конечные устройства могут дополнительно реализовать вариант протокола с изменением до класса С, определенного в настоящем стандарте. Любое конечное устройство должно поддерживать совместимость с классом А.

Сеть LoRa различает базовый класс устройств (класс А) и классы с дополнительными функциями (класс В, класс С) (рисунок 1).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Приложение конечного**  **устройства** | | | | | | Приложение |
| **МАС- уровень** | | | | | | МАС |
| **Класс А**  (базовый) | | | **Класс С**  (дополнительный) | | | Опции МАС |
| **ЛЧМ-модуляция** | | | | | | Модуляция |
| **KZ** | **RU** | **EU** | **EU** | **US** | **…** | Региональные параметры |
| **864** | **868** | **868** | **433** | **915** |

**Рисунок 1 - Классы устройств LoRaWAN**

- Двунаправленные конечные устройства (класс А): Конечные устройства класса А предназначены для обеспечения двунаправленной связи, в которой за каждой передачей сообщения конечным устройством в восходящую линию связи следуют два коротких окна приема для нисходящей линии связи. Интенсивность передачи каждым конечным устройством зависит от его собственной потребности в связи с небольшим отклонением времени начала передачи сообщения на случайную величину (протокол ALOHA). Класс А обеспечивает экономичный расход энергии и подходит в случаях, когда приемлема связь по нисходящей линии связи от сервера только после передачи по восходящей линии связи. Сервер сети для передачи сообщения по нисходящей линии связи должен дождаться следующего сообщения от конечного устройства по восходящей линии связи.

- Двунаправленные конечные устройства с окнами приема по расписанию (класс B) : Устройства класса В открывают дополнительные окна приема по расписанию в дополнение к окнам приема, определенным для устройств класса А. Для синхронизации времени открытия дополнительных окон приема шлюзы излучают маячки (Beacons). Все шлюзы, входящие в состав одной сети, должны излучать маячки одновременно. Маячок содержит идентификатор сети и метку времени в формате всемирного координированного времени (UTC). Использование класса В гарантирует, что при опросе конечных устройств задержка отклика не будет превышать величину, определяемую периодом маячков.

-Двунаправленные конечные устройства с максимальными окнами приема (класс C): У конечных устройств класса С большую часть времени открыто одно из окон приема (за исключением времени передачи сообщения по восходящей линии связи). Конечное устройство класса С будет расходовать больше энергии по сравнению с устройством класса А, но при этом иметь минимальную задержку передачи сообщения от сервера сети к конечному устройству.

**6 Конечные устройства класса А**

**6.1 Формат сообщений на физическом уровне**

**6.1.1 Режимы связи конечного устройства**

По режиму связи конечного устройства различают восходящие и нисходящие сообщения. На физическом уровне сообщения отличаются ортогональной поляризацией и структурой сообщения.

Примечание - Приемопередающее устройство переключается между соответствующими режимами поляризации для отправки восходящего или приема нисходящего сообщения.

**6.1.2 Восходящие сообщения**

Восходящие сообщения отправляются конечными устройствами сетевому серверу (в адрес сетевого сервера) и ретранслируются на сервер одним или несколькими шлюзами (базовыми станциями).

Структура восходящего сообщения на физическом уровне представлена на рисунке 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Преамбула | Физический | Контрольная сумма | Данные | Циклическая |
| (Preamble) | заголовок | физического | (PHYPayload) | контрольная |
|  | (PHDR) | заголовка  (PHDR\_CRC) |  | сумма (CRC) |

**Рисунок 2 - Структура восходящего сообщения на физическом уровне**

Поля «Физический заголовок» (PHDR), «Контрольная сумма физического заголовка» (PHDR\_CRC), и «Циклическая контрольная сумма» (CRC) вычисляются аппаратно.

**6.1.3 Нисходящие сообщения**

Каждое нисходящее сообщение передается сервером сети конечному устройству через один шлюз\*.

Структура нисходящего сообщения на физическом уровне представлена на рисунке 3.

\_\_\_\_\_\_\_\_\_

\* В настоящем стандарте не рассматривается передача многоадресных сообщений сервером сети множеству конечных устройств.

|  |  |  |  |
| --- | --- | --- | --- |
| Преамбула | Физический | Контрольная сумма | Данные |
| (Preamble) | заголовок | физического заголовка | (PHYPayload) |
|  | (PHDR) | (PHDR\_CRC) |  |

**Рисунок 3 - Структура нисходящего сообщения на физическом уровне**

Примечание - Проверка целостности поля «Данные» (PHYPayload) с использованием контрольной суммы не проводится с целью минимизации длины сообщения с минимальным влиянием на любые ограничения рабочего цикла в используемом диапазоне частот.

**6.1.4 Окна приема (класс А)**

После каждой передачи восходящего сообщения конечное устройство должно открыть два коротких окна приема. Момент открытия окон RX1 и RX2 приема нисходящих сообщений задается относительно времени окончания передачи восходящего сообщения приведен на рисунке 4.

**Рисунок 4 - Временные сегменты приема конечного устройства**

**6.1.4.1 Канал, скорость передачи данных и время открытия первого окна приема**

Первое окно приема RX1 использует частоту, значение которой зависит от частоты в восходящей линии связи, и скорость приема данных в окне RX1, которая зависит от скорости передачи данных предыдущего восходящего сообщения. Окно приема RX1 открывается с задержкой RECEIVE\_DELAY1 секунд (+/- 20 микросекунд) после окончания модуляции восходящего сообщения (RECEIVE\_DELAY1 и RECEIVE\_DELAY2 определены в 6.4). Отношение между скоростью передачи данных в восходящей линии связи и скоростью передачи нисходящих данных в окне приема RX1 является региональным параметром и определено в 6.1. По умолчанию, скорость передачи нисходящего сообщения в первом окне приема RX1 равна скорости передачи данных, использованной при передаче последнего восходящего сообщения.

**6.1.4.2 Канал, скорость передачи данных и время открытия второго окна приема**

Второе окно приема RX2 использует фиксировано заданные частоту приема и скорость передачи данных и открывается с задержкой RECEIVE\_DELAY2 секунд (+/- 20 микросекунд) после окончания модуляции восходящего сообщения (RECEIVE\_DELAY1 и RECEIVE\_DELAY2 определены в 6.4). Частота и скорость передачи данных, используемые в окне RX2, могут быть изменены с помощью MAC команд (см. 6.3). Значения частоты и скорости передачи данных по умолчанию являются региональными параметрами и указаны в 6.1.

**6.1.4.3 Длительность окон приема**

Длина окна приема должна быть не менее времени, необходимого радиоприемнику конечного устройства для эффективного распознавания преамбулы нисходящего сообщения.

**6.1.4.4 Активность приемника во время окна приема**

Если поле «Преамбула» (Preamble) обнаружено во время одного из окон приема, то радиоприемник остается активным до окончания демодуляции нисходящего сообщения. Если сообщение распознано и затем демодулировано в пределах первого окна приема RX1, и кадр был адресован этому конечному устройству после проверки адреса и кода целостности сообщения (MIC), то конечное устройство не должно открывать второе окно приема RX2.

**6.1.4.5 Отправка сообщения на конечное устройство**

Если необходимо передать конечному устройству нисходящее сообщение, то сервер сети должен инициировать передачу точно в начале хотя бы одного из двух окон приема. Если нисходящее сообщение передается в течение обоих окон, то должны передаваться идентичные пакеты в каждое окно.

**6.1.4.6 Примечание об окнах приема**

Конечное устройство не должно передавать следующее восходящее сообщение до того, как получит нисходящее сообщение в первом или втором окне приема, открытых по факту предыдущей передачи, или пока не закроется второе окно приема, открытое по факту предыдущей передачи.

**6.1.4.7 Прием или передача по другим протоколам**

Конечное устройство может принимать или передавать данные по другим протоколам в свободные промежутки времени между передачей по протоколу LoRaWAN и окнами приема RX1/RX2 до тех пор, пока устройство не нарушает региональных регуляторных ограничений и требований протокола LoRaWAN.

**6.2 Формат сообщений на МАС-уровне**

Структура восходящего радиосообщения на физическом уровне представлена на рисунке 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Преамбула  (Preamble) | Физический  заголовок  (PHDR) | Контрольная сумма физического заголовка (PHDR\_CRC) | Данные  (PHYPayload) | Циклическая контрольная сумма (CRC) |

**Рисунок 5 - Структура восходящего радиосообщения на физическом уровне**

Структура нисходящего радиосообщения на физическом уровне представлена на рисунке 6.

|  |  |  |  |
| --- | --- | --- | --- |
| Преамбула | Физический | Контрольная сумма | Данные |
| (Preamble) | заголовок | физического заголовка | (PHYPayload) |
|  | (PHDR) | (PHDR\_CRC) |  |

**Рисунок 6 - Структура нисходящего радиосообщения на физическом уровне**

Все восходящие и нисходящие сообщения содержат поле «Данные» (PHYPayload), имеющее один из следующих форматов:

- в соответствии с рисунком 7. Поле начинается с поля «МАС-заголовок» (MHDR) размером 1 байт, затем передается поле «МАС-сообщение» (MACPayload), и заканчивается полем «Код целостности сообщения» (MIC) размером 4 байта.

|  |  |  |
| --- | --- | --- |
| МАС-заголовок  (MHDR) | МАС-сообщение  (MACPayload) | Код целостности  сообщения (MIC) |

**Рисунок 7 - Структура поля «Данные» (PHYPayload) (вариант 1)**

- в соответствии с рисунком 8. Поле начинается с поля «МАС-заголовок» (MHDR) размером 1 байт, затем передается поле «Запрос на присоединение к сети» (Join- Request) или «Запрос на переприсоединение к сети» (Rejoin-Request), и полем «Код целостности сообщения» (MIC) размером 4 байта.

|  |  |  |
| --- | --- | --- |
| МАС- | Запрос на присоединение к сети (Join-Request) или | Код целостности |
| заголовок | Запрос на повторное присоединение к сети | сообщения |
| (MHDR) | (Rejoin-Request) | (MIC) |

**Рисунок 8 - Структура поля «Данные» (PHYPayload) (вариант 2)**

- в соответствии с рисунком 9. Поле начинается с поля «МАС-заголовок» (MHDR) размером 1 байт, затем передается поле «Подтверждение присоединения к сети» (Join-Accept).

|  |  |
| --- | --- |
| МАС-заголовок  (MHDR) | Подтверждение присоединения к сети (Join-Accept) |

**Рисунок 9 - Структура поля «Данные» (PHYPayload) (вариант 3)**

Структура поля «МАС-сообщение» (MACPayload) представлена на рисунке 10.

|  |  |  |
| --- | --- | --- |
| Заголовок МАС-  сообщения (FHDR) | Порт (FPort) | Прикладные данные  (FRMPayload) |

**Рисунок 10 - Структура поля «МАС-сообщение» (MACPayload)**

Структура поля «Заголовок МАС-сообщения» (FHDR) представлена на рисунке 11.

|  |  |  |  |
| --- | --- | --- | --- |
| Короткий адрес конечного устройства (DevAddr) | Управление кадром  (FCtrl) | Счетчик кадров (FCnt) | Параметры кадра (FOpts) |

**Рисунок 11 - Структура поля «Заголовок МАС-сообщения» (FHDR)**

**6.2.1 Поле «Данные» (PHYPayload)**

Структура поля «Данные» (PHYPayload) с указанием размера полей представлена на рисунке 12.

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер (в** | 1 | 7... М | 4 |
| **байтах)** |  |  |  |
| **Данные** | МАС-заголовок | МАС-сообщение | Код целостности |
| **(PHYPayload)** | (MHDR) | (MACPayload) | сообщения (MIC) |

**Рисунок 12 - Структура поля «Данные» (PHYPayload)**

Максимальная длина *(М)* поля «МАС-сообщение» (MACPayload) зависит от скорости передачи сообщения и является региональным параметром.

**6.2.2 Поле «МАС-заголовок» (MHDR)**

Структура поля «МАС-заголовок» (MHDR) с указанием размера полей представлена на рисунке 13.

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер бита** | 7..5 | 4..2 | 1..0 |
| **МАС-заголовок** | Тип сообщения | RFU | Основная версия |
| **(MHDR)** | (МТуре) |  | формата данных (Major) |

**Рисунок 13 - Структура поля «МАС-заголовок» (MHDR)**

**6.2.2.1 Поле «Тип сообщения» (МТуре)**

В протоколе LoRaWAN различают восемь различных типов MAC-сообщений, в соответствии с таблицей 1.

**Таблица 1-Типы МАС-сообщений**

|  |  |
| --- | --- |
| **Значение**  **Поля** | **Описание** |
| 000 | Запрос на присоединение (Join Request) |
| 001 | Подтверждение присоединения (Join Accept) |
| 010 | Неподтверждаемые восходящие данные (Unconfirmed Data Up) |
| 011 | Неподтверждаемые нисходящие данные (Unconfirmed Data Down) |
| 100 | Подтверждаемые восходящие данные (Confirmed Data Up) |
| 101 | Подтверждаемые нисходящие данные (Confirmed Data Down) |
| 110 | Запрос на повторное присоединение (Rejoin Request) |
| 111 | Сообщения собственного протокола (Proprietary protocol messages) |

**6.2.2.1.1** Сообщения «Запрос на присоединение» (Join Request), «Подтверждение присоединения» (Join Accept) и «Запрос на повторное присоединение» (Rejoin Request)

Сообщения: «Запрос на присоединение» (Join Request), «Подтверждение присоединения» (Join Accept) и «Запрос на повторное присоединение» (Rejoin Request) используются в процедуре активации по воздуху, описанной в 6.4.2, и в целях роуминга.

**6.2.2.1.2 Сообщения с данными**

Сообщения с данными используются для передачи MAC команд и данных приложений (прикладных данных), которые могут быть объединены вместе в одном сообщении. Подтвержденное сообщение с данными, требующее уведомления о получении сообщения, должно быть подтверждено получателем, в то время как неподтвержденное сообщение не требует отправки уведомления (подробная временная диаграмма работы механизма подтверждения приведена в Разделе 8). Собственные типы сообщений могут использоваться для реализации нестандартных форматов сообщений, которые не совместимы со стандартными сообщениями, но должны использоваться для поддержки устройств (между устройствами), имеющими общее понимание собственных (нестандартных) расширений. Когда конечное устройство или сетевой сервер получают сообщение неизвестного нестандартного формата, необходимо его проигнорировать.

Целостность сообщения обеспечивается разными способами для разных типов сообщения, что определяется ниже для каждого типа сообщения.

**6.2.2.2 Поле «Основная версия формата данных» (Major)**

Значения поля «Основная версия формата данных» (Major) и их описание представлены в таблице 2.

**Таблица 2 - Значения поля «Основная версия формата данных» (Major)**

|  |  |
| --- | --- |
| **Значения поля** | **Описание** |
| 00 | LoRaWAN R1 |
| 01..11 | RFU |

Примечание - Значения поля «Основная версия формата данных» (Major) определяют формат сообщений, которыми обмениваются в ходе процедуры присоединения к сети (активации) (см. 6.4.2), и первые четыре байта поля «МАС-сообщение» (MACPayload). Для каждой основной версии формата данных конечные устройства могут реализовывать разные неосновные версии формата данных. Неосновная версия, используемая конечным устройством, должна быть известна сетевому серверу до ее использования (например, как часть информации, персонализирующей устройство). Если устройство или сервер сети получают данные неизвестной или неподдерживаемой версии формата данных, то они должны быть проигнорированы.

**6.2.3 Поле «МАС-сообщение» (MACPayload)**

Структура поля «МАС-сообщение» (MACPayload) представлена на рисунке 14 и содержит поля «Заголовок МАС-сообщения» (FHDR), необязательное поле «Порт» (FPort) и необязательное поле «Прикладные данные» (FRMPayload).

|  |  |  |
| --- | --- | --- |
| Заголовок МАС- | Порт | Прикладные данные |
| сообщения (FHDR) | (FPort) | (FRMPayload) |
|  |  |  |

**Рисунок 14-Структура поля «МАС-сообщение» (MACPayload)**

Поле «МАС-сообщение» (MACPayload), состоящее только из поля «Заголовок МАС-сообщения» (FHDR), является корректным.

**6.2.3.1 Поле «Заголовок МАС-сообщения» (FHDR)**

Структура поля «Заголовок МАС-сообщения» (FHDR) с указанием размеров элементов представлена на рисунке 15. При наличии поля «Параметры кадра» (FOpts) для его шифрования должен использоваться ключ шифрования NwkSEncKey в соответствии с 0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер (в байтах)** | 4 | 1 | 2 | 0..15 |
| **Заголовок МАС-** | Короткий адрес | Управление | Счетчик | Параметры |
| **сообщения** | конечного устройства | кадром | кадров | кадра |
| **(FHDR)** | (DevAddr) | (FCtrl) | (FCnt) | (FOpts) |

**Рисунок 15 - Структура поля «Заголовок МАС-сообщения» (FHDR)**

Для нисходящих сообщений поле «Управление кадром» (FCtrl) имеет структуру, указанную на рисунке 16.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Номер бита** | 7 | 6 | 5 | 4 | [3..0] |
| **Управление**  **кадром (FCtrl)** | Адаптивная  скорость  передачи  данных  (ADR) | RFU | Получение  подтверждения  сообщения  (АСК) | Отложенные  кадры  (FPending) | Длина  параметров  кадра  (FOptsLen) |

**Рисунок 16 - Структура поля «Управление кадром» (FCtrl) для нисходящих сообщений**

Для восходящих сообщений поле «Управление кадром» (FCtrl) имеет структуру, указанную на рисунке 17.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Номер бита** | 7 | 6 | 5 | 4 | [3..0] |
| **Управление кадром (FCtrl)** | Адаптивная  скорость передачи данных (ADR) | Запрос подтверждения адаптивной скорости передачи данных (ADRACKReq) | Получение подтверждения сообщения (АСК) | Класс В (ClassB ) | Длина параметров  кадра (FOptsLen) |

**Рисунок 17 - Структура поля «Управление кадром» (FCtrl) для восходящих**

**сообщений**

**6.2.3.1.1** Адаптивное управление скоростью передачи данных (поля «Адаптивная скорость передачи данных» (ADR), «Запрос подтверждения адаптивной скорости передачи данных (ADRACKReq))

LoRa-сеть позволяет конечным устройствам индивидуально настраивать любые из допустимых скоростей передач данных и выходную мощность передатчика. Эта особенность используется в протоколе LoRaWAN для адаптации и оптимизации скорости передачи данных и выходной мощности передатчика стационарных и малоподвижных конечных устройств. Для указания данного свойства используется поле «Адаптивная скорость передачи данных» (ADR) и в этом случае сеть будет оптимизирована для использования самой быстрой возможной скорости передачи данных.

Адаптивное управление скоростью передачи данных становится невозможным, когда затухание сигнала в радио канале быстро и непрерывно меняется. Когда сетевой сервер не в состоянии управлять скоростью передачи данных устройства, управление должно осуществляться на уровне приложения конечного устройства. В этом случае рекомендуется использовать различные скорости передачи данных. Уровень приложения должен всегда минимизировать общее эфирное время с учетом состояния сети.

Если бит поля «Адаптивная скорость передачи данных» (ADR) восходящего сообщения установлен, то сеть будет управлять скоростью передачи данных и выходной мощностью передатчика конечного устройства через соответствующие MAC команды. Если бит поля «Адаптивная скорость передачи данных» (ADR) не установлен, то сеть не будет управлять скоростью передачи данных и выходной мощностью передатчика конечных устройств независимо от качества принимаемого сигнала. Дополнительно, с целью снижения количества потерянных пакетов, сервер сети может посылать команды, изменяющие маску канала или количество повторений для каждого восходящего сообщения.

Если бит поля «Адаптивная скорость передачи данных» (ADR) нисходящего сообщения установлен, то он информирует конечное устройство, что сетевой сервер может посылать ADR-команды. Конечное устройство может устанавливать/снимать бит поля «Адаптивная скорость передачи данных» (ADR) восходящего сообщения.

Если бит поля «Адаптивная скорость передачи данных» (ADR) нисходящего сообщения не установлен, то для конечного устройства это означает, что из-за быстрых изменений в радиоканале сеть временно не может оценить лучшую скорость передачи данных. В этом случае у устройства есть следующий выбор:

- сбросить бит поля «Адаптивная скорость передачи данных» (ADR) восходящего сообщения и управлять своей скоростью передачи данных в восходящей линии связи с использованием собственной стратегии. Эта стратегия должна быть типовой для мобильных конечных устройств;

- игнорировать (сохранить поля «Адаптивная скорость передачи данных» (ADR) восходящего сообщения установленным) и применить нормальную сниженную скорость передачи данных при отсутствии нисходящих ADR-команд. Эта стратегия должна быть типовой для неподвижных конечных устройств.

Бит поля «Адаптивная скорость передачи данных» (ADR) может быть установлен и сброшен конечным устройством или сетью по запросу. Однако, по мере возможности, ADR-схема должна быть включена, чтобы увеличить время автономной работы конечного устройства и увеличить производительность сети.

Примечание - В некоторых случаях мобильные конечные устройства большую часть времени являются неподвижными. Поэтому, в зависимости от состояния мобильности, конечное устройство может запросить сеть оптимизировать скорость передачи данных, используя бит поля «Адаптивная скорость передачи данных» (ADR) восходящего сообщения.

По умолчанию выходная мощность передатчика равна максимальной допустимой мощности передачи для устройства, с учетом ограничений, описанных в Разделе 9. Устройство должно использовать этот уровень мощности, пока не будет запроса об уменьшении от сети через MAC команду LinkADRReq.

Если скорость передачи данных устройства оптимизирована сетью и превышает скорость передачи данных устройства по умолчанию или выходная мощность передатчика ниже чем по умолчанию, то устройство должно периодически проверять, что сеть по-прежнему получает восходящие пакеты. Каждый раз, как увеличивается счетчик восходящих кадров (для каждого нового восходящего сообщения при повторных передачах счетчик не увеличивается), устройство увеличивает значение счетчика ADR\_ACK\_CNT. После превышения счетчиком ADR\_ACK\_CNT (AD R\_AC К\_С NT >= ADR\_ACK\_LIMIT) порогового значения AD R\_AC K\_L IM IT восходящих сообщений без какого-либо ответа сервера сети, устройство устанавливает бит поля «Запрос подтверждения адаптивной скорости передачи данных» (ADRACKReq). Сервер сети должен отреагировать нисходящим кадром в течение следующих ADR\_ACK\_DELAY кадров (восходящих), любой полученный нисходящий пакет сбрасывает счетчик ADR\_ACK\_CNT восходящей линии связи. Бит поля «Подтверждение получения сообщения» (АСК) в нисходящем сообщении устанавливать не нужно, так как любой ответ в окно приема конечного устройства указывает на то, что шлюз (базовая станция) все еще получает восходящие сообщения от этого устройства. Если ответ не будет получен в течение ближайших ADR\_ACK\_DELAY восходящих сообщений (т.е. после превышения количества восходящих сообщений, оставшихся без ответа со стороны сервера сети более чем ADR\_ACK\_LIMIT + ADR\_ACK\_DELAY), то устройство должно попытаться восстановить связь путем наращивания выходной мощности передатчика до значения по умолчанию, если это возможно, и затем переключиться на более низкую скорость передачи данных, что увеличивает дальность связи. Конечное устройство должно продолжать снижать свою скорость передачи данных шаг за шагом, всякий раз при достижении ADR\_ACK\_DELAY. После того, как устройство достигло самой низкой скорости передачи данных, оно должно повторно включить все частотные каналы восходящей линии связи по умолчанию.

Бит поля «Запрос подтверждения адаптивной скорости передачи данных» (ADRACKReq) не должен быть установлен, если устройство использует скорость передачи данных и выходную мощность передатчика по умолчанию, потому что в этом случае никакие действия не могут быть предприняты для улучшения качества связи.

Примечания

1 Отсутствие необходимости немедленного ответа на запрос подтверждения ADR обеспечивает гибкость сети при оптимальном планировании своих передач по нисходящей линии связи.

2 При передаче по восходящей линии связи бит поля «Запрос подтверждения адаптивной скорости передачи данных» (ADRACKReq) устанавливается, если ADR\_ACK\_CNT >= ADR\_ACK\_LIMIT и текущая скорость передачи данных больше, чем определенная для устройства минимальная скорость передачи данных, или мощность передачи меньше, чем по умолчанию, или текущая маска канала использует только подмножество всех каналов по умолчанию. При других условиях он обнуляется.

В таблице 3 приведен пример обратной последовательности снижения скорости передачи данных при условии, что константы ADR\_ACK\_LIMIT=32 и ADR АСК DELAY=32.

**Таблица 3 - Пример обратной последовательности снижения скорости передачи данных**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Значение**  **счетчика**  **ADR\_ACK\_CNT** | **Поле «Запрос подтверждения адаптивной скорости передачи данных» (ADRACKReq)** | **Скорость**  **передачи**  **данных** | **Выходная**  **мощность**  **передатчика**  **(дБм)** | **Маска канала** |
| От 0 до 63 | 0 | SF11 | 9 | Текущий список каналов |
| От 64 до 95 | 1 | SF11 | 9 | Текущий список каналов |
| От 96 до 127 | 1 | SF11 | 14 | Текущий список каналов |
| От 128 до 159 | 1 | SF12 | 14 | Текущий список каналов |
| Более 160 | 0 | SF12 | 14 | Все доступные каналы |

**6.2.3.1.2** Поле «Подтверждение получения сообщения» (АСК) и процедура подтверждения

При получении сообщения, требующего уведомления получения данных, получатель должен ответить сообщением, в котором установлен бит поля «Подтверждение получения сообщения» (АСК). Если отправителем является конечное устройство, сеть попытается отправить подтверждение, используя одно из окон приема, открытое конечным устройством после операции отправки. Если отправителем является шлюз, конечное устройство передает уведомление по своему усмотрению (см. примечание ниже).

Подтверждение отправляется только в ответ на последнее полученное сообщение и никогда не ретранслируется.

Примечание - Конечному устройству разрешено насколько возможно упростить процедуру подтверждения и иметь несколько вариантов ее выполнения. Конечное устройство может передавать явное (возможно пустое) сообщение с подтверждением получения данных сразу после получения сообщения с данными, требующее подтверждения получения. Кроме того, конечное устройство может отложить передачу подтверждения, прикрепив его к следующему сообщению данных.

**6.2.3.1.3 Процедура повторной передачи**

Нисходящий кадр (требующий или не требующий подтверждения от конечного устройства) не должен повторно отправляться сервером сети с использованием одного и того же счетчика значения нисходящих кадров. Если после отправки нисходящего сообщения, требующего подтверждения, сервер сети не получил от устройства уведомление о доставке, то сервер сети должен уведомить об этом сервер приложения. Сервер приложения принимает решение о целесообразности повторной передачи нисходящего сообщения, требующего подтверждения.

Восходящие кадры (требующие и не требующие подтверждения) передаются число раз, равное NbTrans (см. 6.3.3), за исключением, если получено корректное нисходящее сообщение в ходе одной следующей передачи. Параметр NbTrans может использоваться администратором сети, чтобы контролировать избыточность восходящих пакетов узла связи для получения заданного качества обслуживания. Конечное устройство должно выполнять скачкообразное переключение частоты, между повторными передачами. После каждого повторения оно должно ждать, пока не закроется окно приема. Задержка между повторными передачами остается на усмотрение конечного устройства и может быть различной для каждого конечного устройства.

Устройство должно остановить любую дальнейшую ретрансляцию восходящих кадров, требующих подтверждения, если получен соответствующий нисходящий кадр с подтверждением.

Устройства класса В и С должны прекратить любую дальнейшую ретрансляцию восходящих кадров, когда корректное нисходящее сообщение получено в окно приема RX1.

Устройства класса А должны прекратить любую дальнейшую ретрансляцию восходящих кадров, когда корректное нисходящее сообщение получено в окно приема RX1 или RX2.

Если сервер сети получает один и тот же восходящий кадр более NbTrans раз, то это может быть признаком атаки на сервер или неисправности устройства. В этом случае сервер сети не должен обрабатывать избыточные кадры.

Примечание - Сервер сети, обнаружив атаку в виде повторных передач, может принять дополнительные меры, такие как уменьшение параметра NbTrans на 1 или удаление дублирующих восходящих кадров, принятых через один и тот же канал, или другие механизмы.

**6.2.3.1.4 Поле «Отложенные кадры» (FPending), только для нисходящих сообщений**

Поле «Отложенные кадры» (FPending) используется только в нисходящей линии связи и указывает на то, что сеть имеет данные, ожидающие своей отправки, и поэтому запрашивает конечное устройство максимально быстро открыть еще одно окно приема посредством отправки другого восходящего сообщения.

Подробное использование поля «Отложенные кадры» (FPending) описано в 8.3.

**6.2.3.1.5 Поле «Счетчик кадров» (FCnt)**

Каждое конечное устройство имеет три счетчика кадров, чтобы отслеживать и хранить число кадров данных, переданных в восходящую линию связи сетевому серверу (FCntUp), и отправленных в нисходящую линию связи сервером сети на конечное устройство (FCntDown).

В нисходящем направлении существует две разных схемы счетчиков кадров:

- единая схема счетчика, в котором все порты используют один и тот же счетчик кадров FCntDown, когда конечное устройство работает по спецификации LoRaWAN 1.0;

- схема с двумя счетчиками, в которой первый счетчик NFCntDown используется для MAC команд на порт 0 или когда поле FPort отсутствует. Второй счетчик AFCntDown используется для всех других портов, когда устройство работает по спецификации LoRaWAN 1.1.

В схеме с двумя счетчиками, счетчик NFCntDown управляется сервером сети, а счетчик AFCntDown управляется сервером приложений.

Примечание - Спецификации LoRaWAN v1.0 и более ранние версии поддерживают только один счетчик FCntDown (общий по всем портам), и сетевой сервер должен обеспечить поддержку схемы для устройств, работающих по спецификации LoRaWAN с версией до v1.1.

Всякий раз, когда устройство с активацией по воздуху (ОТАА, Over The Air Activation) успешно обрабатывает сообщение подтверждения присоединения (Join Accept), счетчик кадров на устройстве (FCntUp) и счетчики кадров на стороне сети (NFCntDown и AFCntDown) для этого конечного устройства сбрасываются до 0.

Устройства с активацией через персонализацию (АВР, Activation By Personalization) имеют свои счетчики кадров, которые инициализируются в 0 при изготовлении. На устройствах с активацией через персонализацию счетчики кадров не должны сбрасываться на протяжении времени жизни устройства. Если конечное устройства подвергается отключению питания во время срока службы (например, замена аккумуляторной батареи), счетчики кадров должны сохраняться.

Впоследствии счетчик кадров FCntUp увеличивается с каждым восходящим сообщением. Счетчик кадров NFCntDown увеличивается с каждым нисходящим сообщением в случае, когда значение в поле «Порт» (FPort) равно нулю, или данное поле отсутствует. Счетчик кадров AFCntDown увеличивается с каждым нисходящим сообщением при значении поля «Порт» (FPort), отличном от нуля. На стороне получателя, соответствующий счетчик будет синхронизироваться с полученным значением при условии, что полученное значение было увеличено по сравнению с текущим значением счетчика и поле «Код целостности сообщения» (MIC) соответствует значению кода целостности сообщения, вычисленному локально с использованием соответствующего сетевого сеансового ключа. Счетчик FCnt не увеличивается при многократных передачах кадров, требующих и не требующих подтверждения получения (см. параметр NbTrans). Сетевой сервер должен отбрасывать прикладные данные из повторно принятых кадров и передавать серверу приложений только один экземпляр кадра.

Счетчики кадров имеют размерность 32 бита. Поле «Счетчик кадров» (FCnt) кодирует младшие 16 бит 32-х битного счетчика кадров (т.е. счетчика данных FCntUp для кадров данных, передаваемых в восходящую линию связи, и счетчика AFCntDown/NFCntDown для кадров данных, передаваемых по нисходящей линии).

Конечное устройство не должно использовать те же значения счетчика данных FCntUp с одним и тем же сеансовым ключом приложения или сеансовым сетевым ключом, за исключением случаев ретрансляции одного и того же кадра с подтверждением или без подтверждения.

Конечное устройство не должно обрабатывать любые повторные передачи нисходящих кадра. Последующие повторные передачи должны игнорироваться без обработки.

Примечания

1 Это означает, что устройство будет отправлять подтверждение единожды, только после приема нисходящего кадра, требующего подтверждения. Аналогично устройство будет генерировать только одно восходящее сообщение после приема кадра с установленным битом поля «Отложенные кадры» (FPending).

2 Поскольку поле «Счетчик кадров» (FCnt) несет только младшие 16 бит из 32-х бит счетчика кадров, сервер должен вычислить 16 старших битов счетчика кадров из наблюдений за трафиком.

**6.2.3.1.6 Параметры кадра (поле «Длина параметров кадра» (FOptsLen) и поле «Параметры кадра» (FOpts))**

Поле «Длина параметров кадра» (FOptsLen) в поле «Управление кадром» (FCtrl) указывает фактическую длину поля «Параметры кадра» (FOpts), включенного в кадр.

Поле «Параметры кадра» (FOpts) передает MAC команды длиной до 15 байт, которые присоединяются к кадру данных. Список MAC команд приведен в п.6.3.

Если значение поля «Длина параметров кадра» (FOptsLen) равно нулю, то поле «Параметры кадра» (FOpts) должно отсутствовать. Если значение поля «Длина параметров кадра» (FOptsLen) отличается от нуля, т.е. если MAC команды присутствуют в поле «Параметры кадра» (FOpts), то не может быть использовано нулевое значение в поле «Порт» (FPort) (поле должно отсутствовать, или значение отличаться от нуля).

MAC команды не могут одновременно присутствовать в поле «Данные» (FRMPayload) и в поле «Параметры кадра» (FOpts). Если это произойдет, то устройство должно игнорировать кадр.

Если в заголовке кадра имеется поле «Параметры кадра» (FOpts), то поле «Параметры кадра» (FOpts) должно быть зашифровано до того, как будет рассчитано значение поля «Код целостности сообщения» (MIC).

Должна быть использована схема шифрования на основе универсального алгоритма, определенного в стандарте IEEE 802.15.4/2006, приложение В, с использованием шифрования AES\* с длиной ключа 128 бит.

\_\_\_\_\_

\* AES (Advanced Encryption Standard) - симметричный алгоритм блочного шифрования

Ключ К, используемый для поля «Параметры кадра» (FOpts) восходящих и нисходящих сообщений, является сетевым сеансовым ключом шифрования NwkSEncKey.

Шифруемым полем является: pld = FOpts

Для каждого сообщения алгоритм определяет единый блок А, представленный на рисунке 18.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Размер(в байтах)** | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
| ***А*** | 0x01 | 4 х 0x00 | Направление  (Dir) | Короткий  адрес  конечного  устройства  (DevAddr) | Счетчик восходящих кадров (FCntUp) или счетчик нисходящих кадров (NFCntDwn) | 0x00 | 0x00 |

**Рисунок 18 - Структура блока данных для шифрования**

В поле «Направление» (Dir) должно быть установлено значение, равное 0 для восходящих кадров, и равное 1 для нисходящих кадров.

В результате шифрования блока А получается блок S:

S = aesl28\_encrypt(K,А)

Шифрование и расшифрование поля «Параметры кадра» (FOpts) выполняется с усечением:

(*pld* | pad16) xor S

до первых len(*pld*) байт.

**6.2.3.1.7 Поле «Класс В» (Class B)**

Значение поля «Класс В» (ClassB) в восходящих сообщениях устанавливается равным 1 для того, чтобы сообщить сетевому серверу, что устройство переключено в режим класса В и готово принимать нисходящие сообщения проверки связи по расписанию.

**6.2.3.2 Поле «Порт» (FPort)**

Если поле «Прикладные данные» (FRMPayload) не является пустым, то поле «Порт» (FPort) должно присутствовать. При наличии поля «Порт» (FPort) значение, равное 0, указывает, что поле «Прикладные данные» (FRMPayload) содержит только MAC команды, и любые полученные кадры в этом случае будут обработаны на уровне реализации LoRaWAN (список допустимых MAC команд представлен в 6.3). Значения поля «Порт» (FPort) в диапазоне от 1 до 223 (от 0x01 до OxDF) выделены под приложения, и любые полученные кадры в этом случае должны быть доступны на уровне приложения. Значение поля «Порт» (FPort), равное 224, выделено под протокол испытаний LoRaWAN уровня MAC. Реализация LoRaWAN должна стирать любые запросы передачи от уровня приложения, где значение поля «Порт» (FPort) не входит в диапазон от 1 до 224.

Примечание - Значение поля «Порт» (FPort), равное 224, предназначено для сценариев беспроводных испытаний соответствия окончательной версии устройства MAC настоящей спецификации, без необходимости полагаться на определенные испытательные версии устройств для прикладных задач. Испытание не должно совпадать со штатными операциями, но реализация MAC уровня устройства должна быть точно такой, как используется для обычного приложения. Протокол испытаний обычно использует для шифрования AppSKey'. Это гарантирует, что сетевой сервер не сможет включить режим испытания устройства без участия владельца устройства. Способ получения AppSKey приложением испытания на сетевой стороне, если испытание запускается на устройстве, подключенном к рабочей сети, не рассматривается в настоящем стандарте. Способ передачи АррКеу на испытательный стенд для обеспечения процедуры присоединения к сети, если испытание выполняется с использованием активации по воздуху на специальном испытательном стенде (не в рабочей сети), также выходит за рамки настоящего стандарта.

Протокол испытания работает на прикладном уровне и определяется за рамками настоящего стандарта, так как он является протоколом прикладного уровня.

\_\_\_\_\_\_\_\_

Application session key - сессионный ключ размером 128 бит, используемый для шифрования данных на уровне приложения

Значения поля «Порт» (FPort) в диапазоне от 225 до 255 (от 0хЕ1 до OxFF) зарезервированы для будущих стандартизированных расширений приложений.

Размеры элементов поля «МАС-сообщение» (MACPayload) указаны на рисунке 19.

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер (в байтах)** | От 7 до 22 | 0 или 1 | От 0 до *N* |
| **МАС-сообщение**  **(MACPayload)** | Заголовок МАС- сообщения (FHDR) | Порт  (FPort) | Прикладные данные (FRMPayload) |

**Рисунок 19 - Размеры элементов поля «МАС-сообщение»**

N - число байт прикладных данных. Допустимый диапазон N является региональным параметром и определяется в разделе 9. *N* должно быть равным или меньшим, чем:

N< М - 1 - (длина поля «Заголовок МАС-сообщения» (FHDR) в байтах),

где М - максимальная длина данных в поле «МАС-сообщение» (MACPayload).

**6.2.3.3 128-битное шифрование данных в поле «Прикладные данные» (FRMPayload)**

Если кадр данных содержит прикладные данные, то поле «Прикладные данные» (FRMPayload) должно быть зашифровано перед расчетом кода целостности сообщения.

Используемая схема шифрования основана на алгоритме, определенном в стандарте IEEE 802.15.4/2006, приложение В, с использованием шифрования AES с ключом длиной 128 бит.

Используемый для шифрования ключ *К* зависит от значения поля «Порт» (FPort) сообщения данных, в соответствии с таблицей 4.

**Таблица 4 - Значение ключа для шифрования К**

|  |  |  |
| --- | --- | --- |
| **FPort** | **Направление передачи** | ***К*** |
| 0 | Восходящее / Нисходящее | NwkSEncKey |
| От 1 до 255 | Восходящее / Нисходящее | AppSKey |

Шифрованию подлежит поле:

*pld* = FRMPayload

Для каждого сообщения данных алгоритм определяет последовательность блоков *Аi* для *i* = 1...k, где *k* = ceil*(len(pld*)/16), в соответствии с рисунком 20.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Размер (в байтах)** | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
| ***Аi*** | 0x01 | 4 х 0x00 | Направление  (Dir) | Короткий  адрес  конечного  устройства  (DevAddr) | Счетчик кадров на устройстве (FCntUp) или счетчик кадров на стороне сети (NFCntDown) или счетчик кадров на стороне сети (AFCntDown) | 0x00 | *i* |

**Рисунок 20 - Последовательность блоков шифрования**

В поле «Направление» (Dir) устанавливается значение 0 для восходящих кадров и значение 1 для нисходящих кадров.:

Блоки А шифруются, в результате получается последовательность S из блоков *Si*

Si= *aes*l28\_encrypt(K,A*i*) где i = 1 ..к

S = Si | S2 |…|Sk

Шифрование и расшифрование прикладных данных выполняется с усечением:

(pld | pad16) *xor S*

до первых len(pld) байт.

**6.2.4 Поле «Код целостности сообщения» (MIC)**

Поле «Код целостности сообщения» (MIC) вычисляется по всем полям сообщения:

visq = MHDR | FHDR | FPort | FRMPayload

где len(msg) обозначает длину сообщения в байтах.

**6.2.4.1 Нисходящие кадры**

Код целостности сообщения нисходящего кадра рассчитывается по алгоритму AES-CMAC, определенному в документе RFC4493:

*стас* = aesl28\_*cmac*(*SNwkSIntKey*, В0 | *msg*)

MIC = *cmac*[0. .3]

где блок *Во* определяется в соответствии с рисунком 21.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Размер**  **(в**  **байтах)** | 1 | 2 | 2 | 1 | 4 | 4 | 1 | 1 |
| ***Во*** | 0x49 | Счетчик  подтвержденных  восходящих  кадров  (ConfFCnt) | 2 х 0x00 | Направление (Dir) = 0x01 | Короткий  адрес  конечного  устройства  (DevAddr) | Счетчик нисходящих кадров (AFCntDown) или счетчик нисходящих кадров (NFCntDwn) | 0x00 | *len(msg)* |

**Рисунок 21 - Структура блока для вычисления кода целостности сообщения нисходящего кадра**

Если устройство подключено к LoRaWAN 1.1 серверу и поле «Подтверждение получения сообщения» (АСК) для нисходящего кадра установлен, то кадр является уведомлением на предыдущий восходящий кадр с подтверждением получения; значение поля «Счетчик подтвержденных восходящих кадров» (ConfFCnt) равно значению счетчика подтвержденных восходящих кадров по модулю 216. Во всех остальных случаях значения поля равно 0.

В поле «Направление» (Dir) должно быть установлено значение 0x01 для нисходящих кадров от сервера сети.

**6.2.4.2 Восходящие кадры**

Для расчета кода целостности сообщения восходящего кадра определяется блок Во согласно рисунку 22 и блок В1 согласно рисунку 23.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Размер**  **(в**  **байтах)** | 1 | 4 | 1 | 4 | 4 | 1 | 1 |
| ***Во*** | 0x49 | 4x0x00 | Направление  (Dir) | Короткий  адрес  конечного  устройства  (DevAddr) | Счетчик восходящих кадров на устройстве (FCntUp) | 0x00 | *len{msg)* |

**Рисунок 22 - Структура блока *Во* для вычисления кода целостности сообщения**

**восходящего кадра**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Размер (в байтах)** | 1 | 2 | 1 | 1 | 1 | 4 | 4 | 1 | 1 |
| ***Вi*** | 0x49 | Счетчик  подтвержденных  восходящих  кадров  (ConfFCnt) | Скорость передачи данных восходящей линии связи (TxDr) | Индекс  канала  передачи  (TxCh) | Направ­  ление  (Dir) | Короткий  адрес  конечного  устройства  (DevAddr) | Счетчик восходящих кадров на устройстве (FCntUp) | 0x00 | len(msg) |

**Рисунок 23 - Структура блока *Bi*для вычисления кода целостности сообщения**

**восходящего кадра**

Если бит в поле «Подтверждение получения сообщения» (АСК) в восходящем кадре установлен, то кадр является уведомлением на нисходящий кадр с подтверждением получения, тогда значение поля «Счетчик подтвержденных восходящих кадров» (ConfFCnt) равно значению счетчика подтвержденных нисходящих кадров по модулю 216. Во всех остальных случаях значение поля «Счетчик подтвержденных восходящих кадров» (ConfFCnt) должно быть равно 0x0000.

cmacS = aesl28\_cmac(SNwkSIntKey, *В1* | msg)

cmacF = aesl28\_cmac(FNwkSIntKey, *B0* | msg)

Если устройство подключено к серверу, поддерживающему LoRaWAN 1.0, то:

MIC = cmacF [0..3]

Если устройство подключено к серверу, поддерживающему LoRaWAN 1.1, то:

MIC = *cmacS*[0..1] | *cmacF*[0..1]

В поле «Направление» (Dir) должно быть установлено значение 0x00 для восходящих кадров к серверу сети.

**6.3 МАС-команды**

Набор МАС-команд предназначен для сетевого администрирования и может быть использован для обмена между сервером сети и конечным устройством на МАС- уровне, в соответствии с таблицей 5. Команды МАС-уровня не обрабатываются сервером приложений и приложением, запущенным на устройстве.

Один кадр данных может содержать любую последовательность МАС-команд, вставленную в поле «Параметры кадра» (FOpts) или отправленную в отдельном кадре данных, в поле «Прикладные данные» (FRMPayload) с значением поля «Порт» (FPort), равным 0.

МАС-команды, передаваемые в поле «Параметры кадра» (FOpts), отправляются в шифрованном виде и не должны превышать 15 байт. МАС-команды, отправляемые в поле «Прикладные данные» (FRMPayload), всегда шифруются и их длина не должна превышать максимальную длину поля «Прикладные данные» (FRMPayload).

МАС-команда состоит из поля «Идентификатор команды» (CID) размером 1 байт и поля «Атрибуты команды» размером от 0 байт до 14 байт. Для некоторых команд поле «Атрибуты команды» может быть пустым.

МАС-команды со значениями идентификаторов команды (CID) от 0x01 до 0x7F предназначены для использования во всех сетях международной LoRaWAN.

Приемная сторона отвечает / подтверждает получение МАС-команд в том же порядке, как они передаются. Ответ для каждой МАС-команды последовательно добавляется в буфер. На все МАС-команды, полученные в одном кадре, ответы должны быть переданы в одном кадре (т.е. буфер, содержащий ответы на МАС- команды, должен быть отправлен в одном кадре). Если длина буфера с МАС-ответами больше, чем максимальная длина поля «Параметры кадра» (FOpts), устройство должно отправить буфер в поле «Прикладные данные» (FRMPayload) на порт 0. Если устройству надо отправить прикладные данные и MAC ответы, и они не помещаются в один кадр, то MAC ответы должны быть отправлены в первую очередь. Если длина буфера превышает максимальный используемый размер поля «Прикладные данные» (FRMPayload), устройство перед сборкой кадра должно уменьшить буфер до максимального размера поля «Прикладные данные» (FRMPayload). Поэтому ответы на последние МАС-команды могут бьггь неполными. В любом случае, полный список MAC команд выполняется, даже если буфер, содержащий МАС-ответы, должен быть обрезан. Сетевой сервер не должен генерировать последовательность МАС-команд, на которые конечное устройство не может ответить одним восходящим кадром. Сетевой сервер должен вычислять максимальный размер поля «Прикладные данные» (FRMPayload) для ответов на MAC команды, следующим образом:

- если поле «Адаптивная скорость передачи данных» (ADR) последнего восходящего сообщения имеет значение 0, то необходимо устанавливать максимальный размер поля «Прикладные данные» (FRMPayload), соответствующий самой низкой скорости передачи данных;

- если поле «Адаптивная скорость передачи данных» (ADR) последнего восходящего сообщения имеет значение 1, то необходимо устанавливать максимальный размер поля «Прикладные данные» (FRMPayload) соответствующим скорости передачи данных, используемой устройством для передачи последнего восходящего сообщения.

Примечание - При получении обрезанного МАС-ответа сервер сети может ретранслировать МАС-команды, на которые не получил ответ.

**Таблица 5 -МАС-команды**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Команда** | **Передается** | | Краткое описание |
| **конечному**  **устройству** | **базовой**  **станции** |
| 0x01 | *Resetind* | X | — | Используются устройствами с активацией через персонализацию для индикации сброса и согласования версий протокола. |
| 0x01 | *ResetConf* | — | X | Подтверждает команду *Resetind.* |
| 0x02 | *LinkCheckReq* | X | — | Используется конечным устройством для проверки его подключения к сети. |
| 0x02 | *LinkCheckAns* | — | X | Ответ на команду LinkCheckReq.  Содержит оценку мощности полученного сигнала, указывающую на качество приема конечного устройства (устойчивость связи). |
| 0x03 | *LinkADRReq* | — | X | Запрашивает конечное устройство изменить скорость передачи данных, мощность передачи, количество повторений или канал. |
| 0x03 | *LinkADRAns* | X | — | Подтверждает команду LinkADRReq. |
| 0x04 | *DutyCycleReq* | — | X | Устанавливает максимальное агрегированное значение рабочего цикла устройства на передачу. |
| 0x04 | *DutyCycleAns* | X | — | Подтверждает команду DutyCycleReq. |
| 0x05 | *RXParamSetupReq* | — | X | Устанавливает параметры окон приема. |

*Продолжение таблицы 5*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Команда** | **Передается** | | **Краткое описание** |
| **Конечному устройству** | **Базовой станции** |
| 0x05 | *RXParamSetupAns* | X | — | Подтверждает команду *RXParamSetupReq.* |
| 0x06 | *DevStatusReq* | — | X | Запрашивает статус конечного устройства. |
| 0x06 | *DevStatusAns* | X | — | Возвращает статус (состояние) конечного устройства, а именно уровень заряда его батареи и отношение сигнал/шум (оценка демодуляции). |
| 0x07 | *NewChannelReq* | — | X | Создает или изменяет определение радиоканала. |
| 0x07 | *NewChannelAns* | X | — | Подтверждает команду *NewChannelReq.* |
| 0x08 | *RXTimingSetupReq* | — | X | Устанавливает временные интервалы для окон приема. |
| 0x08 | *RXTimingSetupAns* | X | — | Подтверждает команду  *RXTimingSetupReq.* |
| 0x09 | *TxParamSetupReq* | — | X | Используется сервером сети, чтобы установить максимально допустимое время задержки (dwell time) и максимальную эффективную изотропную мощность излучения (EIRP) конечного устройства, на основе локальных соглашений и нормативных актов. |
| 0x09 | *TxParamSetupAns* | X | — | Подтверждает команду *TxParamSetupReq.* |
| ОхОА | *DIChannelReq* | — | X | Изменяет определение нисходящего радиоканала окна приема RX1 путем смещения частоты передачи нисходящей линии от частоты восходящей линии связи (т.е. создание асимметричного канала). |
| ОхОА | *DIChannelAns* | X | — | Подтверждает команду DIChannelReq. |
| *Продолжение таблицы 5* | | | | |
| **CID** | **Команда** | **Передается** | | **Краткое описание** |
| **Конечному устройству** | **Базовой станции** |
| ОхОВ | *Rekeylnd* | X | — | Используется устройством с активацией по воздуху для оповещения об обновлении сеанса связи устройства с сервером сети (обновление ключей). |
| ОхОВ | *RekeyConf* | — | X | Подтверждает команду Rekeylnd |
| ОхОС | *ADRParamSetupReq* | — | X | Используется сервером сети для установки ADR АСК LIMT и ADR\_ACK\_DELAY параметров конечного устройства. |
| ОхОС | *ADRParamSetupAns* | X | — | Подтверждает команду ADRParamSetupReq. |
| OxOD | *DeviceTimeReq* | X | — | Используется конечным устройством для запроса текущей даты и времени. |
| OxOD | *DeviceTimeAns* | — | X | Сеть отправляет ответ на запрос DeviceTimeReq. |
| ОхОЕ | *ForceRejoinReq* | — | X | Посылается сетью для запроса немедленного повторного присоединения (Rejoin) устройства, дополнительно указывается количество и периодичность повторов. |
| ОхОF | *RejoinParamSetupReq* | — | X | Используется сетью для установки периодичности отправки устройством запросов на повторное присоединение (Rejoin). |
| ОхОF | *RejoinParamSetupAns* | X | — | Подтверждает команду RejoinParamSetupReq. |
| От  0x80  до  OxFF | *Проприетарные*  *команды* | X | X | Зарезервировано для команд, действующих только в региональных сетях |

Примечания

1 Как правило, конечное устройство будет отвечать только один раз на любую полученную MAC команду. Если ответ потерян, то сеть вынуждена будет снова послать команду. Сервер сети решает, что команда должна быть отправлена повторно, когда она получает новое восходящее сообщение, которое не содержит ответа.

Только *RxParamSetupReq*, *RxTimingSetupReq* и *DIChannelReq* имеют другой механизм подтверждения, описанный в соответствующих разделах, так как они влияют на параметры нисходящей линии связи.

2 Когда МАС-команда инициируется конечным устройством, сеть делает все возможное для отправки подтверждения/ответа в окна приема RX1/RX2 сразу после запроса. Если ответ не получен в окна RX1 и RX2, то конечное устройство может реализовать любой механизм повтора, который сочтет нужным.

3 Длина МАС-команды не задается явно и должна быть неявно известной по МАС- реализации. Поэтому неизвестные МАС-команды не могут быть пропущены, и первая неизвестная МАС-команда завершает обработку последовательности МАС-команд. Целесообразно использовать МАС-команды, соответствующие версиям стандарта, в котором МАС-команда была впервые опубликована. Таким образом, все МАС-команды, реализованные до появления настоящего стандарта, могут быть обработаны конечным устройством даже среди МАС-команд, описанных только в текущей версии спецификации LoRaWAN и если она новее, чем реализация конечного устройства.

**6.3.1 Команды индикации сброса (Resetind, ResetConf)**

Данная МАС-команда доступна только для устройств с активацией через персонализацию, активированных в сети с сервером сети, поддерживающим LoRaWAN 1.1. На сервере сети, поддерживающим только LoRaWAN 1.0, данная МАС- команда не реализована.

Устройства с активацией по воздуху не должны отправлять эту команду. Сетевой сервер должен игнорировать команду *Resetind*, поступившую от устройства с активацией по воздуху.

С помощью команды *Resetind*, устройство с активацией через персонализацию извещает сети, что оно было повторно инициализировано, и что оно переключено на свои MAC и радио настройки по умолчанию (те. параметры, изначально запрограммированные в устройстве при изготовлении, за исключением трех счетчиков кадров). Команда *Resetind* должна добавляться в поле «Параметры кадра» *(FOpts)* всех восходящих кадров, пока не будет получена команда *ResetConf*.

Данная команда не является сигналом сетевому серверу, что были сброшены счетчики кадров. Счетчики кадров нисходящих и восходящих сообщений не должны сбрасываться в устройствах с активацией через персонализацию.

Примечание - Данная команда предназначена для устройств с активацией через персонализацию, питание которых может быть отключено в какой-то момент времени (например, замена батареи). Устройство может потерять настройки соединения на сеансном уровне, хранящиеся в ОЗУ (кроме счетчиков кадров, которые должны быть сохранены в энергонезависимой памяти). В этом случае, устройство нуждается в том, чтобы как-то сообщить серверу сети о потере настроек соединения на сеансном уровне. В будущих версиях протокола LoRaWAN, эта команда может также использоваться для согласования некоторых параметров протокола между устройством и сервером сети.

Команда *Resetlnd* включает в себя дополнительный номер версии LoRaWAN, поддерживаемой устройством , в соответствии с рисунком 24 и рисунком 25.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **Данные *Resetlnd* *(Resetlnd* Payload)** | Версия LoRaWAN устройства (Dev LoRaWAN version) |

**Рисунок 24 - Структура команды *Resetlnd***

|  |  |  |
| --- | --- | --- |
| **Биты** | 7:4 | 3:0 |
| **Версия LoRaWAN устройства** | RFU | Дополнительный номер |
| **(Dev LoRaWAN version)** |  | версии (Minor)=1 |

**Рисунок 25 - Структура команды *Resetlnd***

Поле «Дополнительный номер версии» (Minor) указывает на дополнительный номер версии LoRaWAN, поддерживаемую конечным устройством, в соответствии с таблицей 6.

**Таблица 6-Значения поля «Дополнительный номер версии» (Minor)**

|  |  |
| --- | --- |
| **Дополнительный номер версии (Minor)** | **Значение**  **поля** |
| RFU | 0 |
| 1 (LoRaWAN x.1) | 1 |
| RFU | От 2 до 15 |

Когда сетевой сервер получает *Resetlnd*, он отвечает командой *ResetConf*. Команда *ResetConf* содержит один байт данных, закодированных в соответствии с версией LoRaWAN, поддерживаемой сервером сети с использованием формата, соответствующего версии LoRaWAN устройства (Dev LoRaWAN version), в соответствии с рисунком 26.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **Данные *ResetConf (*ResetConf *Payload*)** | Версия LoRaWAN сервера сети (Serv LoRaWAN version) |

**Рисунок 26 - Структура команды *ResetConf***

Версия сервера, которую несет *ResetConf*, должна совпадать с версией устройства. Любое другое значение является недопустимым.

Если версия сервера не совпадает с версией устройства, устройство должно отбросить команду *ResetConf* и повторно отправить команду *Resetind* в следующем восходящем кадре.

**6.3.2 Команды проверки подключения к сети (*LinkCheckReq, LinkCheckAns*)**

С помощью команды *LinkCheckReq* конечное устройство может проверить свое подключение к сети. Команда не имеет полезных данных.

Когда сетевой сервер получает *LinkCheckReq* через один или несколько шлюзов, он отвечает командой *LinkCheckAns*. Структура команды *LinkCheckAns* представлена на рисунке 27.

|  |  |  |
| --- | --- | --- |
| **Размер (в байтах)** | 1 | 1 |
| **Данные *LinkCheckAns*** | Устойчивость | Число шлюзов |
| ***(LinkCheckAns* Payload)** | демодуляции (Margin) | (GwCnt) |

**Рисунок 27 - Структура команды *LinkCheckAns***

Поле «Устойчивость демодуляции» (Margin) представляет собой 8-битовое целое число без знака в диапазоне от 0 до 254 и указывает значение устойчивости связи в дБ, полученное по факту успешного приема последней МАС-команды *LinkCheckReq*. Значение, равное 0, означает, что кадр был получен на минимальном уровне демодуляции (0 дБ или отсутствии значения), а значение, равное 20, например, означает, что кадр достиг шлюза с 20 дБ запаса относительно порога демодуляции. Значение, равное 255, зарезервировано для будущего использования.

Поле «Число шлюзов» (GwCnt) определяет число шлюзов (базовых станций), которые успешно получили последнюю команду *LinkCheckReq*.

Значения минимального уровня соотношения сигнал/шум для демодуляции пакета представлены в таблице 7.

**Таблица 7 -Значения минимального уровня соотношения сигнал/шум для демодуляции пакета**

|  |  |
| --- | --- |
| **Скорость передачи** | **Сигнал/шум (SNR) минимального уровня демодуляции пакета (дБ)** |
| DR0 | -20 |
| DR1 | -17,5 |
| DR2 | -15 |
| DR3 | -12,5 |
| DR4 | -10 |
| DR5 | -7,5 |

**6.3.3 Адаптация скорости передачи данных (*LinkADRReq, LinkADRAns*)**

С помощью команды *LinkADRReq* сетевой сервер запрашивает конечное устройство выполнить адаптацию скорости передачи данных. Структура команды представлена на рисунке 28 и рисунке 29.

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер (в байтах)** | 1 | 2 | 1 |
| **Данные *LinkADRReq (LinkADRReq Payload)*** | Запрошенная скорость передачи данных и выходная мощность передатчика (*DataRate\_TXPower*) | Маска  канала  (*ChMask*) | Избыточность  (*Redundancy*) |

**Рисунок 28 - Структура команда *LinkADRReq***

|  |  |  |
| --- | --- | --- |
| **Биты** | [7:4] | [3:0] |
| **Запрошенная скорость передачи данных и выходная мощность передатчика (DataRate\_TXPower)** | Запрошенная скорость передачи данных (*DataRate*) | Выходная мощность передатчика (*TXPower*) |

**Рисунок 29 - Структура команды *LinkADRReq***

Запрошенная скорость передачи данных (*DataRate*) и выходная мощность передатчика (*TXPower*) являются региональными параметрами и определены в Разделе 9. Выходная мощность передатчика, указанная в команде, должна рассматриваться, как максимальная мощность передачи, на которой может работать устройство. Если в команде задана мощность, превышающая максимальную мощность устройства, то конечное устройство должно подтвердить успешное выполнение команды и работать на своей максимально возможной мощности.

Значение *OxF* (15 в десятичном формате) запрошенной скорости передачи данных (*DataRate*) или выходной мощности передатчика (*TXPower*) означает, что устройство должно игнорировать это поле и сохранить текущие значение параметра. Маска канала (*ChMask*) кодирует использование каналов для передачи в восходящей линии связи (бит 0 соответствует младшему разряду) в соответствии с Таблицей 8.

**Таблица 8 - Кодировка поля «Маска канала» (*ChMask*)**

|  |  |
| --- | --- |
| **Биты** | **Используемые**  **каналы** |
| 0 | Канал 1 |
| 1 | Канал 2 |
| .. | .. |
| 15 | Канал 16 |

Бит в поле «Маска канала» (ChMask), установленный в 1, означает, что соответствующий канал может использоваться для передачи в восходящей линии связи, если этот канал обеспечивает скорость передачи данных, в настоящее время используемую устройством. Бит, установленный в 0, означает, что соответствующие каналы следует исключить.

Поле «Избыточность» (*Redundancy*) (Рисунок 30) включает поле «Число передач» (*NbTrans*). Это поле используется для восходящих кадров, требующих и не требующих подтверждения получения. Значением по умолчанию является 1, которое соответствует одной передаче каждого кадра. Допустимый диапазон от 1 до 15. Если получено значение поля «Число передач» (*NbTrans*), равное 0, то устройство должно сохранить текущее значение числа передач неизменным.

|  |  |  |  |
| --- | --- | --- | --- |
| **Биты** | 7 | I6:4] | [3:0] |
| **Избыточность** | RFU | Управление маской | Число передач |
| **(Redundancy)** |  | канала (ChMaskCntl) | (NbTrans) |

**Рисунок 30 - Структура поля «Избыточность» (*Redundancy*)**

Поле «Управления маской канала» (*ChMaskCntl*) управляет интерпретацией ранее определенного битового поля «Маска канала» (*ChMask*). Поле «Управления маской канала» (*ChMaskCntl*) контролирует блок из 16 каналов, к которым применяется поле «Маска канала» (*ChMask*). Оно также может быть использовано, чтобы глобально включить или выключить все каналы. Использование этого атрибута является региональным параметром и определяется в разделе 9.

Сетевой сервер может включить несколько последовательных команд *LinkAdrReq* в одно нисходящее сообщение. С целью конфигурирования маски канала конечного устройства, конечное устройство должно обработать всю последовательность *LinkAdrReq* в сообщении, в том порядке, в котором они переданы в нисходящем сообщении, как один единый блок команд. Сетевой сервер не должен включать в нисходящее сообщение более одного такого блока команд. Конечное устройство должно послать одну команду *LinkAdrAns*, чтобы подтвердить или отклонить весь единый *ADR* командный блок. Если нисходящее сообщение несет более одного единого *ADR* командного блока, то устройство должно обработать только первый из них и отправить *NAck* (команда *UnkADRAns* со всеми битами состояния, установленными в 0), в ответ на все остальные *ADR* командные блоки. Устройство должно обрабатывать поля «Скорость передачи данных» (*DataRate*), «Выходная мощность передатчика» (*TXPower*) и «Число передач» (*NbTrans*) только из последней команды *LinkAdrReq* в последовательности *ADR* командного блока, так как значения этих параметров определяют общее состояние устройства. В поле «Получение подтверждения сообщения» (*АСК*) бит маски канала в ответе должен отражать принятие/отклонение окончательного плана каналов после обработки всех элементов управления маской канала в последовательности *ADR* командного блока.

Частота каждого канала задается для конкретного региона и определена в Разделе 9. Конечное устройство отвечает на *LinkADRReq* командой *UnkADRAns*. Структура команды *UnkADRAns* представлена на рисунке 31 и рисунке 32.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **Данные *UnkADRAns* *(UnkADRAns* Payload)** | Статус (Status) |

**Рисунок 31 - Структура команды UnkADRAns**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Биты** | [7:3] | 2 | 1 | 0 |
| **Статус** | RFU | Выходная | Запрошенная | Маска канала в |
| **(Status)** |  | мощность | скорость передачи в | поле «Получение |
|  |  | передатчика в | поле «Получение | подтверждения |
|  |  | поле «Получение | подтверждение | сообщения» |
|  |  | подтверждение | сообщения» (Data | (Channel mask |
|  |  | сообщения» | rate АСК) | АСК) |
|  |  | (Power АСК) |  |  |

Рисунок 32 - Структура команды UnkADRAns

Биты поля «Статус» (Status) UnkADRAns имеют значения согласно таблице 9.

**Таблица 9 - Кодировка поля «Статус» (Status) UnkADRAns**

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Бит = 0** | **Бит = 1** |
| Маска канала в поле  «Получение подтверждение сообщения» (Channel mask АСК) | Отправленная маска канала разрешает использование пока неопределенного канала или маска канала требует отключения всех каналов. Команда была отклонена и состояние конечного устройства не было изменено. | Отправленная маска канала успешно интерпретирована. Статусы всех в настоящее время определенных каналов были установлены в соответствии с маской |
| Запрошенная скорость передачи в поле  «Получение подтверждение сообщения» данных (Data rate АСК) | Запрошенная скорость передачи данных неизвестна конечному устройству или невозможно обеспечить заданную маску канала (не поддерживается ни одним из включенных каналов). Команда была отклонена, и состояние конечного устройства не было изменено | Скорость передачи данных была успешно установлена или поле «Запрошенная скорость передачи» (DataRate) в запросе было установлено в 15, и он был проигнорирован |
| Выходная мощность передатчика в поле  «Получение подтверждение сообщения» (Power АСК) | Устройство не может работать на уровне или ниже заданного уровня выходной мощности передатчика. Команда была отклонена, и состояние конечного устройства не было изменено | Устройство может работать на уровне или ниже заданного уровня выходной мощности передатчика, или поле «Выходная мощность передатчика» (TXPower) в запросе было установлено в 15, и он был проигнорирован |

Если любой из этих трех битов равен 0, то это значит, что команда не выполнена, и устройство сохранило прежнее состояние.

**6.3.4 Рабочий цикл устройства (DutyCycleReq, DutyCycleAns)**

Команда DutyCycleReq используется сервером сети, чтобы ограничить конечному устройству время на передачу сообщений в радиоэфире. Совокупное ограничение для всех частот соответствует ограничению для каждой используемой частоты. Структура команды представлена на рисунке 33 и рисунке 34.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **Данные *DutyCycleReq (DutyCycleReq* Payload)** | Рабочий цикл передачи (DutyCyclePL) |

**Рисунок 33 - Структура команды DutyCycleReq**

|  |  |  |
| --- | --- | --- |
| **Биты** | 7:4 | 3:0 |
| **Рабочий цикл** | RFU | Максимально допустимый |
| **передачи** |  | рабочий цикл передачи |
| **(DutyCyclePL)** |  | (MaxDCycle) |

**Рисунок 34 - Структура команды *DutyCycleReq***

Максимально допустимый рабочий цикл передачи вычисляется:

*aggregated duty cycle* =

Допустимый диапазон для *MaxDutyCycle* от 0 до 15. Значение 0 соответствует «нет ограничений», если в региональных настройках не указано иначе.

Конечное устройство отвечает на *DutyCycleReq* командой *DutyCycleAns*. MAC команда-ответ *DutyCycleAns* не содержит никаких полезных данных.

**6.3.5 Параметры окон приема (*RXParamSetupReq, RXParamSetupAns*)**

Команда *RXParamSetupReq* позволяет изменять частоту и скорость передачи данных, установленные для второго окна приема (RX2) после каждого восходящего сообщения. Эта команда также позволяет запрограммировать смещение скорости передачи данных в восходящей линии связи относительно скорости передачи данных в нисходящей линии связи в окно приема RX1. Структура команды приведена на рисунке 35 и рисунке 36.

|  |  |  |
| --- | --- | --- |
| **Размер (в байтах)** | 1 | 3 |
| **Данные *RXParamSetupReq (RXParamSetupReq* Payload)** | DLsettings | Frequency |

**Рисунок 35 - Структура команды RXParamSetupReq**

|  |  |  |  |
| --- | --- | --- | --- |
| **Биты** | 7 | [6:4] | [3:0] |
| **DLsettings** | RFU | RXIDRoffset | RX2DataRate |

**Рисунок 36 - Структура команды RXParamSetupReq**

Поле *RXIDRoffset* задает смещение между скоростью передачи данных восходящей линии связи и скоростью передачи данных нисходящей линии связи, использованной для связи с конечным устройством в первом окне приема (RX1). По умолчанию это смещение равно 0. Смещение используется, чтобы учитывать максимальные ограничения плотности покрытия базовых станций в некоторых регионах и балансировать загруженностью восходящей и нисходящей линий радиосвязи. Подробнее о значениях *RX1 DRoffset* описано в Разделе 9.

Поле RX2*DataRate* определяет скорость передачи данных нисходящей линии связи, используемой во втором окне приема RX2, следуя правилам, аналогичным для команды UnkADRReq (0 означает DR0/125кГц, к примеру).

Поле *Frequency* соответствует частоте, используемой для второго окна приема RX2, при этом частота кодируется аналогично описанию команды *NewChannelReq*.

Команда *RXParamSetupAns* используется конечным устройством, чтобы подтвердить прием команды *RXParamSetupReq*. Команда *RXParamSetupAns* должна добавляться в поле «Параметры кадра» (FOpts) всех восходящих сообщений, пока конечное устройство не получит нисходящие сообщение в окно RX1 или RX2 (с длительностью RX2, характерной для устройства класса А). Это гарантирует, что даже при наличии потери восходящих пакетов, сеть всегда в курсе параметров нисходящей линии связи, используемых конечным устройством.

Данные включают однобайтовое поле «Статус» (Status), в соответствии с рисунком 37.

|  |  |
| --- | --- |
| Размер (в байтах) | 1 |
| RXParamSetupAns Payload | Status |

**Рисунок 37 - Структура поля «Статус» (Status)**

Биты поля «Статус» (Status) имеют значение согласно рисунку 38 и таблице 10.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Биты** | 7:3 | 2 | 1 | 0 |
| **Статус (Status)** | RFU | RX1 DRoffset АСК | RX2 Data rate АСК | Channel АСК |

**Рисунок 38 - Структура поля «Статус» (Status)**

Таблица 10 - Кодировка поля «Статус» (Status)

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Бит = 0** | **Бит = 1** |
| Channel АСК | Запрашиваемая частота не применима для конечного устройства | Канал для окна приема RX2 успешно установлен |
| RX2 Data rate АСК | Запрошенная скорость передачи данных неизвестна конечному устройству | Скорость передачи для окна приема RX2 успешно установлена |
| RX1 DRoffset АСК | Смещение между скоростью передачи данных восходящей линии связи и скоростью передачи данных нисходящей линии связи для окна приема RX1 за пределами разрешенного диапазона | Смещение RX1 DRoffset успешно установлено |

Если любой из этих трех битов равен 0, то команда не выполнена, и устройство сохраняет прежнее состояние.

**6.3.6 Статус конечного устройства (DevStatusReq, DevStatusAns)**

С помощью команды *DevStatusReq* сетевой сервер может запросить информацию о состоянии конечного устройства. Команда не имеет атрибутов. Если конечное устройство получило *DevStatusReq*, оно должно ответить командой *DevStatusAns*. Структура команды представлена на рисунке 39.

|  |  |  |
| --- | --- | --- |
| **Размер (в байтах)** | 1 | 1 |
| **DevStatusAns Payload** | Battery | Margin |

**Рисунок 39 - Структура команды DevStatusReq**

Уровень заряда батареи (*Battery*) кодируется в соответствии с Таблицей 11.

**Таблица 11-Кодировка поля «Уровень заряда батареи» (*Battery*)**

|  |  |
| --- | --- |
| **Уровень**  **заряда**  **батареи** | **Описание** |
| 0 | Конечное устройство подключено к внешнему источнику питания |
| От 1 до 254 | Уровень заряда батареи:  1 - находится на минимуме и  254 - находится на максимуме |
| 255 | Конечное устройство не смогло измерить уровень заряда батареи |

Поле *Margin* содержит соотношение сигнал/шум (в дБ), измеренное при приеме последней команды - *DevStatusReq*. Значение Margin округляется до ближайшего целого значения. Это целое 6-ти битовое число со знаком с минимальным значением минус 32дБ и максимальным значением плюс 31 дБ.

|  |  |  |
| --- | --- | --- |
| **Биты** | [7:6] | [5.0] |
| **Status** | RFU | Margin |

**Рисунок 40 - Статус конечного устройства**

**6.3.7 Создание/модификация канала (NewChannelReq, NewChannelAns, DIChannelReq, DIChannelAns)**

Устройства, работающие в регионе, для которого определен фиксированный частотный план каналов не должны выполнять эти МАС-команды (т е. устройство не должно отвечать на команды). Задаваемые каналы должны соответствовать требованиям региональных параметров, описанных в Разделе 9.

Команда NewChannelReq может использоваться для изменения параметров существующего двунаправленного канала или создания нового. Команда задает центральную частоту нового канала и диапазон скоростей передачи данных в восходящей линии, которые могут использоваться на этом канале, в соответствии с рисунком 41.

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер (в байтах)** | 1 | 3 | 1 |
| **NewChannelReq** | Индекс каналов | Частота (Freq) | Диапазон |
| **Payload** | (Chlndex) |  | скоростей |
|  |  |  | передачи данных |
|  |  |  | (DrRange) |

**Рисунок 41 - Структура команды NewChannelReq**

Индекс каналов (*Chlndex*) - это индекс вновь созданного или измененного канала. Для каждого региона (см. раздел 9) устанавливаются каналы «по умолчанию», которые не могут быть изменены с помощью команды *NewChannelReq*.

Если число каналов «по умолчанию» равно N, то нумероваться каналы «по умолчанию» будут от 0 до [N-1], а от N до 15 будут нумероваться редактируемые каналы. Таким образом, индекс каналов (*Chlndex*) может принимать значение от N до 15. Устройство должно быть в состоянии обрабатывать по меньшей мере 16 различных каналов. В определенном регионе устройство может хранить параметры более 16 каналов.

Поле *Freq* (частота) - это 24-битовое целое число без знака. Фактическая частота канала в Гц считается 100 х *Freq*, где значения, представляющие частоты ниже 100 МГц, зарезервированы для использования в будущем. Это позволяет устанавливать частоту канала в диапазоне от 100 МГц до 1,67 ГГц с шагом 100 Гц. Значение *Freq*, равное 0, отключает канал. Конечное устройство должно проверить, что частота на самом деле разрешена (обеспечивается) на аппаратном уровне радиомодуля и вернуть сообщение об ошибке в противном случае.

Поле *DrRange* определяет диапазон скоростей передачи данных для восходящей линии связи, разрешенный для данного канала. Поле разделено на два 4-х битных индекса:

|  |  |  |
| --- | --- | --- |
| **Биты** | [7:4] | [3:0] |
| **DrRange** | MaxDR | MinDR |

**Рисунок 42 - Структура поля «Диапазон скоростей передачи данных» (DrRange)**

В соответствии с соглашениями, определенными в 6.3.3, поле *MinDR* обозначает самый низкий уровень скорости передачи данных для восходящей линии связи, допустимый на этом канале. Например, используя европейские региональные параметры, 0 обозначает DRO в полосе 125 кГц. Аналогичным образом, поле *MaxDR* обозначает самый высокий уровень скорости передачи данных для восходящей линии связи, допустимый на этом канале. Например, DrRange = 0 x77 означает, что только 50 кбит/с GFSK допускается на канале и DrRange = 0x50 означает, что поддерживаются скорости передачи данных от DR0 в полосе 125 кГц до DR5 в полосе 125 кГц.

Измененный канал включается и сразу может быть использован для взаимодействия с сервером.

В окне приема RX1 частота в нисходящей линии устанавливается равной частоте в восходящей линии связи.

Конечное устройство подтверждает получение NewChannelReq, отправкой в ответ команды NewChannelAns. Эта команда содержит атрибут, в соответствии с рисунком 43.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **NewChannelAns Payload** | Status |

**Рисунок 43 - Атрибут команды NewChannelAns**

Биты поля Status имеют значение согласно рисунку 44 и таблице 12.

|  |  |  |  |
| --- | --- | --- | --- |
| **Биты** | 7:2 | 1 | 0 |
| **Status** | RFU | Data rate range ok | Channel frequency ok |

**Рисунок 44 - Значения битов поля Status**

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Бит = 0** | **Бит = 1** |
| Data rate range ok | Указанный диапазон скоростей передачи данных превышает поддерживаемый данным конечным устройством | Диапазон скоростей передачи данных совместим с возможностями конечного устройства |
| Channel frequency ok | Устройство не может использовать данную частоту | Устройство имеет возможность использовать эту частоту |

**Таблица 12 - Кодировка поля Status**

Если любой из этих 2 битов равен 0, то команда не выполнена и новый канал не создан.

Команда *DIChannelReq* позволяет сети ассоциировать различные частоты нисходящей линии связи с окном приема RX1. Эта команда применяется для всех спецификаций физического уровня, поддерживающих команду *NewChannelReq* (поддерживается для региональных параметров Европы (EU) и Китая, и не поддерживается для США и Австралии).

Команда задает центральную частоту, используемую для нисходящей линии связи в окне приема RX1, согласно рисунку 45.

|  |  |  |
| --- | --- | --- |
| **Размер (в байтах)** | 1 | 3 |
| **DIChannelReq Payload** | Индекс | Частота (Freq) |
|  | канала |  |
|  | (Chlndex) |  |

**Рисунок 45 - Структура команды *DIChannelReq***

Поле *Chlndex* - индекс канала, для которого изменяется частота нисходящей линии связи.

Поле *Freq* (частота) - это 24-битовое целое число без знака. Фактическая частота канала в Гц считается 100 х *Freq*, где значения, представляющие частоты ниже 100 МГц зарезервированы для использования в будущем. Это позволяет устанавливать частоту канала в диапазоне от 100 МГц до 1,67 ГГц с шагом 100 Гц. Значение Freq, равное 0, отключает канал. Конечное устройство должно проверить, что частота на самом деле поддерживается на аппаратном уровне радиомодулем, и вернуть сообщение об ошибке в противном случае.

Конечное устройство подтверждает получение *DIChannelReq* отправкой в ответ команды *DIChannelAns*. Команда *DIChannelAns* должна добавляться в поле FOpts всех восходящих сообщений, пока конечное устройство не получит нисходящий пакет. Это гарантирует, что даже при наличии потери пакетов в восходящей линии связи, сеть всегда будет в курсе частот, используемых конечным устройством в нисходящей линии связи. Эта команда содержит атрибут согласно рисунку 46.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **DIChannelAns Payload** | Status |

**Рисунок 46 - Атрибут команды DIChannelAns**

|  |  |  |  |
| --- | --- | --- | --- |
| **Биты** | [7:2] | 1 | 0 |
| **Status** | RFU | Uplink frequency | Channel frequency |
|  |  | exists | ok |

Биты поля *Status* имеют значение согласно рисунку 47 и таблице 13.

**Рисунок 47 - Биты поля Status**

|  |  |  |
| --- | --- | --- |
| **Атрибут** | **Бит = 0** | **Бит = 1** |
| Channel frequency ok | Устройство не может использовать эту частоту | Устройство имеет возможность использовать эту частоту |
| Uplink  frequency  exists | Частота восходящей линии связи не определена для данного канала, частота нисходящей линии связи может быть установлена только для канала, который уже имеет допустимую частоту восходящей линии связи. | Частота восходящей линии  связи для канала  допустима  (корректна) |

**Таблица 13 -Кодировка поля Status**

**6.3.8 Настройка задержки между ТХ и RX (*RXTimingSetupReq, RXTimingSetupAns*)**

Команда *RXTimingSetupReq* позволяет настраивать задержку между окончанием передачи восходящего сообщения (ТХ) и открытие первого окна приема (RX1). Второе окно приема (RX2) открывается через 1 (одну) секунду после первого окна приема. Атрибут команды представлен на рисунке 48

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **RXTimingSetupReq** | Settings |
| **Payload** |  |

**Рисунок 48 - Атрибут команды RXTimingSetupReq**

Поле Delay определяет время задержки. Поле разделено на два 4-разрядных индекса в соответствии с рисунком 49.

|  |  |  |
| --- | --- | --- |
| **Биты** | [7:4] | [3:0] |
| **Settings** | RFU | Del |

**Рисунок 49 - Структура поля Delay**

Задержка (Del) - указывается в секундах. Значение Del, равное 0, соответствует 1 сек.

|  |  |
| --- | --- |
| **Del** | **Задержка (сек.)** |
| 0 | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| … | … |
| 15 | 15 |

**Таблица 14 - Кодировка поля Delay**

Конечное устройство отвечает на команду *RXTimingSetupReq* отправкой команды *RXTimingSetupAns* без атрибутов.

Команда *RXTimingSetupAns* должна добавляться в поле FOpts всех восходящих сообщений, пока конечное устройство не получит нисходящий пакет в окно RX1 или RX2 (с длительностью RX2, характерной для устройства класса А). Это гарантирует, что даже при наличии потери пакетов в восходящей линии связи, сеть будет всегда в курсе параметров нисходящей линии связи, используемых конечным устройством.

**6.3.9 Параметры передачи конечного устройства (TxParamSetupReq, TxParamS etupAns)**

Эта МАС-команда должна выполняться с соблюдением региональных параметров.

Команда *TxParamSetupReq* может быть использована для уведомления конечного устройства о максимально допустимом времени задержки (dwell time), т.е. максимальном времени непрерывной передачи пакета по радиоэфиру, а также максимально допустимой эффективной изотропной мощности излучения (EIRP) конечного устройства. Атрибут команды представлен на рисунке 50.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **TxParamSetupReq** | EIRP\_DwellTime |
| **Payload** |  |

**Рисунок 50 - Атрибут команды TxParamSetupReq**

Структура поля EIRP\_DwellTime описана в рисунке 51.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Биты** | 7:6 | 5 | 4 | 3:0 |
| **EIRP DwellTime** | RFU | DownlinkDwellTime | UplinkDwellTime | MaxEIRP |

**Рисунок 51 - Структура поля EIRP\_DwellTime**

Биты [0...3] команды *TxParamSetupReq* кодируют максимальное значение мощности излучения - MaxEIRP. Значения MaxEIRP охватывают широкий диапазон, включающий параметры всех возможных регионов. При установлении MaxEIRP следует соблюдать региональные параметры, приведенные в разделе 9. MaxEIRP кодируется значениями согласно рисунку 51.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Кодируемое**  **значение** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| **MaxEIRP (dBm)** | 8 | 10 | 12 | 13 | 14 | 16 | 18 | 20 | 21 | 24 | 26 | 27 | 29 | 30 | 33 | 36 |

**Рисунок 51 - Кодировка поля MaxEIRP**

MaxEIRP соответствует верхней границе выходной мощности передатчика устройства. Устройство не обязано передавать на этой мощности, но никогда не должно превышать указанное значение MaxEIRP.

Биты UplinkDwellTime и DownlinkDwellTime определяют максимальное время задержки в восходящей и нисходящей линий связи соответственно, которые кодируются согласно таблице 15.

**Таблица 15 - Кодирование значений задержки**

|  |  |
| --- | --- |
| **Кодируемое**  **значение** | **Время задержки** |
| 0 | Не ограничено |
| 1 | 400 мс |

Если в соответствии с региональными параметрами данная MAC команда поддерживается, то, конечное устройство отвечает на команду *TxParamSetupReq,* отправкой команды *TxParamSetupAns*. Ответ *TxParamSetupAns* не содержит атрибутов.

Если региональные параметры не допускают использование данной MAC команды, то устройство не обрабатывает ее и не передает подтверждения.

**6.3.10 Оповещение об обновлении ключей (*Rekeylnd, RekeyConf*)**

Эта МАС-команда доступна только для ОТА-устройств, активированных в сети с сервером сети, поддерживающим LoRaWAN 1.1. Сервер сети, поддерживающий только LoRaWAN 1.0, не реализует эту МАС-команду.

АВР-устройства не должны выполнять эту команду. Сетевой сервер должен игнорировать команду *Rekeylnd*, поступившую от АВР-устройства.

Для ОТА-устройств МАС-команда *Rekeylnd* используется для подтверждения обновления ключей безопасности и в будущих версиях LoRaWAN (>1.1), чтобы договориться о используемой между конечным устройством и сервером сети версии протокола LoRaWAN. Команда не является оповещением о сбросе (перезагрузке) параметров MAC и радиоканала (см.6.4.2.3).

Команда *Rekeylnd* включает в себя номер младшей версии LoRaWAN, поддерживаемой конечным устройством (согласно рисунку 52 и рисунку 53).

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| ***Rekeylnd* Payload** | Dev LoRaWAN version |

**Рисунок 52 - Структура команды Rekeylnd**

|  |  |  |
| --- | --- | --- |
| **Биты** | [7:4] | [3:0] |
| **Dev LoRaWAN version** | RFU | Minor=1 |

**Рисунок 53 - Структура команды Rekeylnd**

Поле Minor оповещает о неосновной версии LoRaWAN, поддерживаемой конечным устройством, в соответствии с таблицей 16.

**Таблица 16- Кодировка значений поля Minor**

|  |  |
| --- | --- |
| **Minor version** | **Minor** |
| RFU | 0 |
| 1 (LoRaWAN x.1) | 1 |
| RFU | 2:15 |

ОТА-устройства должны отправлять команду *Rekeylnd* во всех восходящих сообщениях, требующих и не требующих подтверждения, после успешной обработки *JoinAccept* (новые сеансовые ключи были получены), пока не будет получена команда *RekeyConf*. Если устройство не получило *RekeyConf* в течение первых ADR\_ACK\_LIMIT восходящих сообщений, то оно должна вернуться в состояние присоединения к сети (Join state). Команды Rekeylnd, присланные от этих устройств после этого, должны быть проигнорированы сервером сети. Сервер сети должен отклонить все восходящие кадры, защищенные новыми сессионными ключами, которые были получены после отправки *JoinAccept* и до первого восходящего кадра, который несет в себе команду *Rekeylnd*.

Когда сетевой сервер получает *Rekeylnd*, он отвечает командой *RekeyConf*. Команда RekeyConf содержит один байт полезных данных с закодированной версией LoRaWAN, поддерживаемой сервером сети, используется тот же формат, что и для «Dev LoRaWAN version», в соответствии с рисунком 54.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| ***RekeyConf* Payload** | Serv LoRaWAN |
|  | version |

**Рисунок 54 - Структура команды RekeyConf**

Версия протокола сервера должна быть больше, чем 0 (0 не допускается), и меньше либо равна версии протокола устройства. Поэтому для устройства, поддерживающего настоящий стандарт в объеме международной спецификации LoRaWAN 1.1, единственным допустимым значением является 1. Если версия сервера недействительна, устройство должно отбросить команду *RekeyConf* и повторно отправить (ретранслировать) команду *Rekeylnd* в следующем восходящем кадре.

**6.3.11 Параметры ADR (ADRParamSetupReq, ADRParamSetupAns)**

Команда *ADRParamSetupReq* позволяет изменять параметры ADR\_ACK\_LIMIT и ADR\_ACK\_DELAY, определенные в алгоритме обратного переключения ADR. Команда ADRParamSetupReq содержит атрибут согласно рисунку 55 и рисунку 56.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **ADRParamSetupReq** | ADRparam |
| **Payload** |  |

**Рисунок 55 - Атрибут команды ADRParamSetupReq**

|  |  |  |
| --- | --- | --- |
| **Биты** | [7:4] | [3:0] |
| **ADRparam** | Limit exp | Delay exp |

**Рисунок 56 - Атрибут команды ADRParamSetupReq**

Поле Limit\_exp устанавливает значение параметра ADR\_ACK\_LIMIT.

ADR\_ACK\_LIMIT = 2Limit\_exp

Допустимый диапазон значений Limit\_exp находится в пределах от 0 до 15, что соответствует диапазону значений от 1 до 32768 для параметра ADR\_ACK\_LIMIT. Поле Delay\_exp устанавливает значение параметра ADR\_ACK\_DELAY.

A D R\_AC K\_D Е LAY = 2Delay-exp

Допустимый диапазон значений Delay\_exp находится в пределах от 0 до 15, что соответствует диапазону значений от 1 до 32768 для параметра ADR\_ACK\_DELAY.

Команда *ADRParamSetupAns* используется конечным устройством, чтобы подтвердить прием команды *ADRParamSetupReq*. Команда *ADRParamSetupAns* не имеет поля данных.

**6.3.12 Время устройства (DeviceTimeReq, DeviceTimeAns)**

Данная МАС-команда доступна, только если устройство активировано в сети с сервером сети, поддерживающим LoRaWAN 1.1. Сервер сети, поддерживающий только LoRaWAN 1.0, не реализует эту MAC команду.

С помощью команды *DeviceTimeReq*, конечное устройство может запрашивать у сети текущие дату и время сети. Запрос не имеет никаких полезных данных.

С помощью команды *DeviceTimeAns*, сетевой сервер предоставляет конечному устройству дату и время сети. Предоставленное время - это время сети, зафиксированное в конце передачи восходящего сообщения. Команда имеет 5 байт полезных данных, заданных согласно рисунку 57.

|  |  |  |
| --- | --- | --- |
| **Размер (в байтах)** | 4 | 1 |
| **DeviceTimeAns Payload** | 32-битное целое без знака: число секунд с начала эпохи\* | 8-битное целое без знака: доли секунды с шагом в 1/28 секунды |

**Рисунок 57 - Структура команды DeviceTimeAns**

Время, предоставленное сетью, должно иметь точность не хуже +/-100 мС.

Примечание - В качестве начальной точки отсчета времени эпохи используется 6 января 1980 года полночь. Поле «секунды» - это количество секунд, прошедшее с момента начала эпохи. Это поле монотонно увеличивается каждую секунду на 1. Чтобы преобразовать это поле в UTC время, високосные секунды должны быть приняты во внимание.

***Пример*** - пятница, 12 февраля 2016 года, 14:24:31 по Гринвичу соответствует 1139322288 секунд с начала эпохи по шкале GPS. По состоянию на июнь 2017, время GPS на 17 секунд впереди времени UTC.

**6.3.13 Вынужденное повторное присоединение к сети (ForceRejoinReq)**

С помощью команды ForceRejoinReq, сеть запрашивает у устройства немедленно передать сообщение с запросом повторного присоединения 0-го или 2-го типа (Rejoin- Request type 0 or type 2) с установленным числом и периодичностью попыток присоединения, и скоростью передачи данных. Этот восходящий RejoinReq может быть использован сетью для немедленной замены ключей устройства или для инициирования процедуры передачи устройства в другой сервер сети в роуминге.

Команда имеет два байта полезных данных. Параметры команды представлены на рисунке 58.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Биты** | [15:14] | [13:11] | [10:8] | 7 | [64] | [3:0] |
| **ForceRejoinReq** | RFU | Period | Max Retries | RFU | RejoinType | DR |

**Рисунок 58 - Структура команды ForceRejoinReq**

Параметры кодируются следующим образом:

Период - задержка между повторами передачи, должна быть равна:

Период = 32 • 2Penod + Rand32

где Rand32 - это псевдо-случайное число в диапазоне [0 .. 32].

Max\_Retries - общее количество попыток, которые выполнит устройство, чтобы отправить запрос на повторное присоединение к сети (Rejoin Request).

- 0: запрос на повторное присоединение к сети будет отправлен только 1 раз (без повтора).

-1: запрос на повторное присоединение к сети должен быть отправлен 2 раза в общей сложности (1+1 повтор).

-…

- 7: запрос на повторное присоединение к сети должен быть отправлен 8 раз (1 + 7 повторов).

Поле RejoinType указывает тип запроса RejoinRequest, который будет передан устройством:

- О или 1: должен быть передан запрос RejoinRequest типа 0.

- 2: должен быть передан запрос RejoinRequest типа 2.

-3 ... 7: зарезервированы для последующего использования.

DR-кадр с RejoinRequest должен бьггь передан с указанной скоростью передачи данных. Соотношение между реальной физической скоростью передачи данных (обусловленной типом используемой модуляции) и значением скорости передачи данных определяется теми же правилами, что и для команды *LinkADRReq* и определяется в соответстви с региональными параметрами (см раздел 9).

Команда не имеет ответа, так как устройство должно отправить RejoinRequest при получении команды. Первая передача сообщения с RejoinReq должна быть осуществлена сразу же после приема команды (но сеть может не получить его). Если устройство получает новую команду *ForceRejoinReq* прежде чем оно достигнет максимального числа повторных передач, то устройство должно продолжить передачу RejoinReq с новыми параметрами.

**6.3.14 Параметры повторного присоединения к сети (RejoinParamSetupReq, RejoinParamSetupAns)**

С помощью команды RejoinParamSetupReq, сеть может запросить устройство периодически отправлять сообщение с запросом на повторное присоединение к сети типа 0 (RejoinRequest type 0) с заданной периодичностью отправки, определенной как время или число восходящих кадров.

И время, и число кадров предлагаются для использования устройствами, которые могут не иметь возможности измерять время. Заданная периодичность устанавливает максимальное время и количество восходящих кадров между двумя отправками RejoinReq. Устройство может передавать RejoinReq чаще заданной периодичности.

Команда имеет единственный байт полезных данных, согласно рисунку 59.

|  |  |  |
| --- | --- | --- |
| **Биты** | [7:4] | [3:0] |
| **RejoinParamSetupReq** | MaxTimeN | MaxCountN |

**Рисунок 59 - Структура команды RejoinParamSetupReq**

Параметры определены следующим образом:

MaxCountN = С = от 0 до 15.

Устройство должно отправлять RejoinRequest типа 0 не реже чем каждое 2С+4 исходящее сообщение.

MaxTimeN = Т = от 0 до 15.

Устройство должно отправлять RejoinRequest типа 0 не реже чем каждые 2Т+10 секунд.

- Т = 0 соответствует примерно 17-ти минутам.

- Т = 15- около 1 -го года.

Устройство должно обеспечивать повторное присоединение к сети по достижении порогового значения количества восходящих кадров. Периодичность повторного присоединения, основанная на времени, является необязательной. Устройство, которое не может реализовать подсчет временного интервала, должно известить об этом в ответе. Ответ содержит один байт полезных данных, согласно рисунку 60.

|  |  |  |
| --- | --- | --- |
| **Биты** | 7:1 | 0 |
| **RejoinParamSetupReq** | RFU | TimeOK |

**Рисунок 60 - Структура команды RejoinParamSetupReq**

Если бит 0 равен 1, то прибор принял установку периодичности в формате времени и количества восходящих кадров, в противном случае он принимает установку периодичности только в виде ограничения количества восходящих кадров.

**6.4 Активация конечного устройства**

Для работы в сетях LoRaWAN каждое конечное устройство должно быть зарегистрировано и активировано.

Активация конечного устройства может быть выполнена двумя способами: по воздуху (*Over The Air Activation*, OTAA) или через персонализацию (*Activation By Personalization*, АВР).

6**.4.1 Данные об активации, сохраняемые в устройстве**

**6.4.1.1 Перед Активацией**

**6.4.1.1.1 JoinEUI**

*JoinEUI* является глобальным идентификатором приложения в IEEE EUI64 адресного пространства, который однозначно идентифицируется сервером присоединения к сети (Join Server), который обеспечивает выполнение процедуры присоединения к сети и производства сеансовых ключей.

Для устройств, поддерживающих активацию по воздуху, *JoinEUI* должен быть сохранен в энергонезависимую память устройства до начала выполнения процедуры присоединения.

Для устройств, поддерживающих только активацию через персонализацию, наличие *JoinEUI* не требуется.

**6.4.1.1.2 DevEUI**

*DevEUI* - это глобальный идентификатор конечного устройства в IEEE EUI64 адресном пространстве, который однозначно идентифицирует конечное устройство.

DevEUI рекомендуется использовать сетевым серверам в качестве уникального идентификатора устройства, независимо от используемого метода активации устройства, для идентификации устройства при межсетевом роуминге (перемещении устройства из одного сегмента сети в другой).

Для устройств, поддерживающих активацию по воздуху, *DevEUI* должен бьгть сохранен в энергонезависимую память устройства до начала выполнения процедуры присоединения.

Для устройств, поддерживающих только активацию через персонализацию, DevEUI не требуется сохранять в энергонезависимую память устройства, но рекомендуется это сделать.

Примечание - Рекомендуется использовать DevEUI также в качестве этикетки устройства (номенклатурного номера) и для управления устройством (учета и сопровождения).

**6.4.1.1.3 Первичные ключи устройства (АррКеу и NwkKey)**

Ключи NwkKey и АррКеу являются первичными ключами AES-128 шифрования, специфицированные для конкретного конечного устройства, назначенные конечному устройству при его производстве. В ходе процедуры присоединения конечного устройства к сети методом активации по воздуху NwkKey используется для получения сеансовых ключей FNwkSlntKey, SNwkSlntKey и NwkSEncKey; и АррКеу используется для получения сеансового ключа AppSKey.

Примечание - При работе с сетевым сервером LoRaWAN v1.1, для получения сеансового ключа приложения используется только АррКеу, поэтому NwkKey может быть использован в сети для управления процедурой присоединения без возможности оператора сети расшифровать данные приложений.

Сервер сети должен быть обеспечен системой безопасности с целью защиты хранения и использования первичных ключей NwkKey и АррКеу. Элементы обеспечивающей системы безопасности могут включать программные и/или программно-аппаратные решения. Требования к системе безопасности выходят за рамки настоящего документа и должны рассматриваться отдельно.

Для обеспечения обратной совместимости с LoraWAN 1.0 и более ранних версий сетевых серверов, которые не поддерживают два первичных ключа, конечные устройства по умолчанию должны быть настроены на использование при присоединении к сети схемы с одним первичным ключом. Признаком такого состояния конечного устройства является установленный в ноль бит №7 «OptNeg» в поле DLsetting в JOIN ACCEPT сообщении.

Устройство в этом случае должно:

- использовать NwkKey для получения сеансовых ключей AppSKey и FNwkSlntKey, как указано в спецификации LoRaWAN 1.0.

- установить SNwkSlntKey и NwkSEncKey равными FNwkSlntKey. Один и тот же сетевой сеансовый ключ используется для расчета кода целостности (MIC) восходящих и нисходящих сообщений, и шифрования кадров данных (MACPayload), в соответствии с спецификацией LoRaWAN 1.0.

NwkKey должен быть сохранен в энергонезависимой памяти устройства, поддерживающего использование процедуры ОТАА.

NwkKey не требуется конечным устройствам, поддерживающим только процедуру АВР.

АррКеу должен бьггь сохранен в энергонезависимой памяти устройства, поддерживающего использование процедуры ОТАА.

АррКеу не требуется конечным устройствам, поддерживающим только процедуру АВР.

NwkKey и АррКеу должны храниться таким образом, чтобы исключить возможность их извлечения и повторного использования злоумышленниками.

Примечание - Поскольку все конечные устройства оснащены уникальными для каждого конечного устройства первичными ключами АррКеу и NwkKey, то, в случае извлечения AppKey/NwkKey из конечного устройства, компрометируется только это одно конечное устройство.

**6.4.1.1.4 Формирование JSIntKey и JSEncKey**

Для ОТА устройств из первичного ключа NwkKey формируются два специальных ключа на весь жизненный цикл устройства:

- JSIntKey используется для формирования MIC в сообщении с запросом Rejoin¬Request первого типа и ответа Join-Accept.

- JSEncKey используется для шифрования Join-Accept, сформированного в ответ на запрос Rejoin-Request.

JSIntKey = aesl28\_encrypt(NwkKey,0x06 | DevEUI | pad16)

JSEncKey = aesl28\_encrypt(NwkKey,0x05 | DevEUI | pad16)

**6.4.1.2 После активации**

После активации в конечном устройстве хранятся следующие дополнительные сведения:

- короткий адрес конечного устройства (DevAddr);

- три сетевых сеансовых ключа (NwkSEncKey/SNwkSlntKey/FNwkSlntKey);

- сеансовый ключ приложения (AppSKey).

**6.4.1.2.1 Короткий адрес конечного устройства (DevAddr)**

*DevAddr* состоит из 32 битов, идентифицирующих конечное устройство в текущей (существующей) сети. Его формат представлен на рисунке 61.

|  |  |  |
| --- | --- | --- |
| **№ Бита** | От 31 до 32-N | От 31 -N до 0 |
| **DevAddr** | AddrPrefix | NwkAddr |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\* N - целое число в диапазоне [7:24].

**Рисунок 61 - Структура DevAddr**

Протокол LoRaWAN поддерживает различные типы сетевых адресов с разным размером адресного пространства. Переменный размер поля AddrPrefix является производным от уникального идентификатора сервера сети *NetID* (см. 6.4.2.3), за исключением значений AddrPrefix зарезервированных для экспериментальных/частных сетей. Поле AddrPrefix позволяет сетевым серверам обнаруживать и в реальном времени управлять устройствами в роуминге. Устройства, которые не соблюдают это правило не смогут переподключаться между двумя сетями, т.к. будет невозможно найти их домашний сетевой сервер.

Младшие (32-N) бита DevAddr - это сетевой адрес конечного устройства (NwkAddr), который может назначаться по усмотрению администратора сети.

Значения AddrPrefix, указанные на рисунке 62, могут бьггь использованы любой частной/экспериментальной сетью и не будут взаимодействовать в роуминге.

|  |
| --- |
| AddrPrefix зарезервированные для частных/экспериментальных сетей |
| **N=7** |
| **AddrPrefix = 7'b0000000 или AddrPrefix = 7’b0000001** |
| NwkAddr - это 25 бит, назначаются по усмотрению администратора сети |

**Рисунок 62 - Значения AddrPrefix**

**6.4.1.2.2 Сетевой сеансовый ключ целостности восходящих сообщений (FNwkSlntKey)**

FNwkSlntKey (forwarding network session integrity key) - это сетевой сеансовый ключ, определенный для конечного устройства. Он используется конечным устройством для вычисления MIC или части MIC (код целостности сообщения) всех восходящих сообщений с данными для обеспечения их целостности, как описано в 6.2.4.

FNwkSlntKey должен храниться таким образом, чтобы предотвратить извлечение и повторное использование злоумышленниками.

**6.4.1.2.3 Сетевой сеансовый ключ целостности нисходящих сообщений (SNwkSlntKey)**

S*NwkSlntKey* (serving network session integrity key) - это сетевой сеансовый ключ, определенный для конечного устройства. Он используется конечным устройством для проверки MIC (код целостности сообщения) всех нисходящих сообщений с данными для обеспечения целостности данных и вычисления половины *MIC* восходящих сообщений.

Примечание - Для расчета MIC восходящих сообщений используется два ключа (FNwkSlntKey и SNwkSlntKey) для того, чтобы обеспечить переадресацию сетевого сервера в настройках роуминга, и обеспечить возможность проверять только половину поля MIC.

Когда устройство подключается к сети LoRaWAN 1.0, сетевой сервер использует один и тот же ключ для расчета MIC как восходящих, так и нисходящих сообщений, как указано в 6.2.4, в этом случае S*NwkSlntKey* принимает значение, совпадающее с *FNwkSlntKey*.

S*NwkSlntKey* должен храниться таким образом, чтобы предотвратить извлечение и повторное использование злоумышленниками.

**6.4.1.2.4 Сетевой сеансовый ключ шифрования (NwkSEncKey)**

NwkSEncKey (network session encryption key) - это сетевой сеансовый ключ, определенный для конечного устройства. Он используется для шифрования и расшифрования восходящих и нисходящих МАС-команд, передаваемых в поле прикладных данных FRMPayload на порт 0 или в поле FOpt. Когда устройство подключается к сети LoRaWAN 1.0, сервер использует один и тот же ключ для шифрования МАС-сообщения (MACPayload) и расчета MIC. В этом случае NwkSEncKey принимает значение, совпадающее с *FNwkSlntKey*.

NwkSEncKey должен храниться таким образом, чтобы предотвратить извлечение и повторное использование злоумышленниками.

**6.4.1.2.5 Сеансовый ключ приложения (AppSKey)**

*AppSKey (application session key)* - это сеансовый ключ приложения, определенный для конечного устройства. Он используется сервером приложений и конечным устройством для шифрования и расшифрования поля прикладных данных (FRMPayload) в информационных сообщениях приложения. Прикладные данные шифруются один раз для передачи между конечным устройством и сервером приложений, при этом их целостность контролируется на каждом переходе: один раз на переходе между конечным устройством и сетевым сервером, другой раз на переходе от сетевого сервера к серверу приложений. Это означает, что вредоносный сетевой сервер может изменять содержание данных во время передачи сообщений, что может помочь сетевому серверу извлечь некоторую информацию о данных, наблюдая за реакцией приложения и конечного устройства на измененные данные. Сетевые серверы рассматриваются как надежные, но приложения, желающие реализовать сквозную конфиденциальность и защитить целостность своих данных, могут использовать дополнительные решения по обеспечению сквозной безопасности.

*AppSKey* должен храниться таким образом, чтобы предотвратить извлечение и повторное использование злоумышленниками.

**6.4.1.2.6 Криптографический контекст сеанса связи**

Криптографический контекст сеанса связи разделяется на сетевой сеанс и прикладной сеанс.

В рамках сетевого сеанса определяются следующие сущности:

- FNwkSlntKey/SNwkSlntKey - сетевые сеансовые ключи целостности.

- NwkSEncKey - сетевой сеансовый ключ шифрования.

- FCntUp - счетчик восходящих по сети кадров.

- FCntDwn (LW 1.0) и NFCntDown (LW 1.1)- счетчики нисходящих по сети кадров.

- DevAddr - адрес конечного устройства.

В рамках сеанса приложения определяются следующие сущности:

- AppSKey - сеансовый ключ приложения.

- FCntUp - счетчик восходящих кадров.

- FCntDown (LW 1.0) и AFCntDown (LW 1.1) - счетчики нисходящих кадров.

Состояние сетевого сеанса поддерживается сервером сети и конечным

устройством. Состояние сеанса приложения поддерживается сервером приложений и конечным устройством.

По окончании процедуры авторизации ОТАА или АВР устанавливается новый безопасный сеанс связи между сервером сети (или сервером приложений) и конечным устройством. Ключи и адрес конечного устройства являются фиксированными в течение всего срока сеанса связи (FNwkSlntKey, SNwkSlntKey, AppSKey, DevAddr). Счетчики кадров увеличиваются по факту обмена пакетами во течение сеанса связи (FCntUp, FCntDwn, NFCntDwn, AFCntDown).

Для устройств ОТАА, счетчики кадров не должны повторно использоваться при одних и тех же сеансовых ключах, поэтому новый сеанс должен быть установлен до момента переполнения счетчиков кадров.

Рекомендуется, чтобы состояние сеанса сохранялось даже при переключении (выключении и включении) питания на конечном устройстве. Невыполнение этого требования для устройств ОТАА означает, что процедуру активации придется выполнять при каждом выключении и включении устройства.

**6.4.2 Активация по воздуху (ОТАА)**

Для активации по воздуху, конечное устройства должно пройти процедуру присоединения к сети, прежде чем участвовать в обмене данными с сетевым сервером. Конечное устройство должно пройти через новую процедуру присоединения к сети каждый раз, при потере сведений о параметрах сеансного уровня связи.

Прежде чем начнется процедура присоединения к сети необходимо, чтобы конечное устройство было персонализировано следующей информацией: DevEUI, JoinEUI, NwkKey и АррКеу.

Примечание - Для активации по воздуху устройства не персонализируются парой сетевых сеансовых ключей. Вместо этого, всякий раз, когда конечное устройство присоединяется к сети, для шифрования и проверки целостности передаваемых на сетевом уровне данных формируются сетевые сеансовые ключи специфические для данного устройства. Таким образом, роуминг конечных устройств между сетями различных провайдеров облегчается. Использование разных сетевых сеансовых ключей и сеансового ключа приложения позволяет дополнительно ограничить сетевые сервера, на которых данные приложений не должны быть прочитаны или изменены сетевым провайдером.

**6.4.2.1 Процедура присоединения к сети**

Со стороны конечного устройства процедура присоединения к сети представляет отправку конечным устройством запроса на присоединение (*Join request*) или повторное присоединение *(Rejoin request*) и получение подтверждения присоединения (*Join accept).*

**6.4.2.2 Сообщение с запросом на присоединение (Join request)**

Процедура присоединения всегда инициируется конечным устройством, путем отправки сообщения с запросом на присоединение. Структура сообщения представлена на рисунке 63.

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер (в байтах)** | 8 | 8 | 2 |
| **Запрос на присоединение** | JoinEUI | DevEUI | DevNonce |

**Рисунок 63 - Структура сообщения с запросом на присоединение**

Сообщение с запросом на присоединение содержит *JoinEUI* и *DevEUI* конечного устройства с последующим полем случайного значения из 2 байт (*DevNonce*).

*DevNonce* - это счетчик, начинается с 0, когда устройство изначально включается и увеличивается с каждым JoinRequest. Значение DevNonce никогда не должно использоваться повторно для заданного значения JoinEUI. Если конечное устройство может быть выключено и включено, то DevNonce не должен изменяться (должен сохраняться в энергонезависимой памяти). Сброс DevNonce без изменения JoinEUI вызовет отклонение сетевым сервером запроса устройства на присоединение к сети. Для каждого конечного устройства, сетевой сервер отслеживает значения DevNonce, использованные конечным устройством, и игнорирует запросы на присоединение к сети, если *DevNonce* не изменился (не увеличился).

Примечание - Этот механизм предотвращает атаки повторного воспроизведения путем отправки ранее записанных сообщений - запросов на присоединение с целью отключения соответствующего конечного устройства от сети. Сетевой сервер в любое время обработает запрос на присоединение и сформирует пакет с подтверждением присоединения, он должен поддерживать и старые параметры криптографического контекста сеанса (ключи и счетчики, если они есть) и новые, пока не получит первый успешный пакет из восходящей линии связи, содержащий *Rekeylnd* команду с использованием настроек нового сеанса. После этого настройки старого сеанса могут быть безопасно удалены.

Значение кода целостности сообщения (*MIC*) для сообщения с запросом на присоединение рассчитывается следующим образом:

*cmac = aesl28\_cmac(NwkKey, MHDR | JoinEUI | DevEUI | DevNonce)*

*MIC = стас[0..3]*

Сообщение запроса на присоединение не шифруется. Сообщение запроса на присоединение может передаваться на любой скорости передачи данных и частоте, случайным образом выбранной из возможных для присоединения каналов. Рекомендуется использовать различные скорости передачи данных. Интервалы между передачами запросов на присоединение должны соблюдать условия, описанные в 6.5. Для каждой следующей передачи запроса на подключение устройство должно увеличить значение DevNonce.

**6.4.2.3 Сообщение с подтверждением соединения (Join-accept)**

Сетевой сервер отвечает на сообщение с запросом на присоединение (или повторное присоединение) - сообщением с подтверждением соединения, если конечному устройству разрешено присоединение к сети. Сообщение с подтверждением соединения отправляется как обычное нисходящее сообщение, но использует задержки JOIN\_ACCEPT\_DELAY1 или JOIN\_ACCEPT\_DELAY2 (вместо RECEIVE\_DELAY1 и RECEIVE\_DELAY2, соответственно). Частота канала и скорость передачи данных, используемых для получения этих двух окон приема идентичны тем, которые используются для окон приема RX1 и RX2.

Ответ конечному устройству не передается, если запрос на подключение не принят.

Сообщение с подтверждением соединения содержит поле случайного значения (*JoinNonce*) из 3 байт, сетевой идентификатор (*NetID*), адрес конечного устройства (*DevAddr*), (*DLSettings*) поле с параметрами нисходящей линии связи, задержку между ТХ и RX (*RxDelay*) и дополнительный список сетевых параметров (CFList & CFListType) для сети, к которой присоединилось конечное устройство. Дополнительные поля CFList & CFListType являются региональными настройками и определены в разделе 9.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Размер (в байтах)** | 3 | 3 | 4 | 1 | 1 | 16 опц. |
| **Подтверждение** | JoinNonce | Home\_NetlD | DevAddr | DLSettings | RxDelay | CFList |
| **соединения** |  |  |  |  |  |

**Рисунок 64 - Структура сообщения с подтверждением соединения**

*JoinNonce* содержит значение счетчика повторных присоединений для конкретного устройства (значения счетчика никогда не повторяются), предоставленное сервером присоединения (join server) и используемое конечным устройством для получения сеансовых ключей *FNwkSlntKey*, *SNwkSlntKey, NwkSEncKey* и *AppSKey.* JoinNonce увеличивается с каждым сообщением с подтверждением присоединения (JoinAccept).

Устройство отслеживает значение JoinNonce, использованное в последнем успешно обработанном JoinAccept (соответствует последней успешной генерации ключей). Устройство принимает JoinAccept только если в поле MIC корректное значение и JoinNonce строго больше, чем записанное ранее. В этом случае новое значение JoinNonce заменяет ранее сохраненное.

Если устройство подвергается периодическому выключению/включению питания JoinNonce, при этом, меняться не должен (должен сохраняться в энергонезависимой памяти).

Уникальный идентификатор сети (*NetID*) составляет 24 бита, за исключением следующих значений *NetID*, отведенных для экспериментальных/частных сетей, управление которыми не осуществляется.

Выделяется 215 зарезервированных значений NetID для частных /экспериментальных сетей, формируемых согласно рисунку 65.

|  |  |  |
| --- | --- | --- |
| 3 | 14 | 7 |
| 3b000 | хххххххххххххх  Произвольное 14-битное значение | 7’b0000000 или 7’b0000001 |

**Рисунок 65 - Формирование значений NetID для частных /экспериментальных**

**сетей**

Поле *Home\_NetlD* в JoinAccept соответствует *NetID* домашней сети устройства. Сеть, которая присваивает DevAddr, и домашняя сеть могут быть разными в состоянии роуминга.

Поле *DLsettings* содержит конфигурацию нисходящей линии связи согласно рисунку 66.

|  |  |  |  |
| --- | --- | --- | --- |
| **Биты** | 7 | 6:4 | 3:0 |
| **DLsettings** | OptNeg | RX1 DRoffset | RX2 Data rate |

**Рисунок 66 - Структура поля DLsettings**

OptNeg бит указывает, какую версию протокола реализует сетевой сервер: LoRaWAN 1.0 (бит не установлен), 1.1 и выше (бит установлен).

Когда бит OptNeg установлен, то:

- будет использоваться протокол версии LoRaWAN 1.1 (или более поздний), соглашение между конечным устройством и сетевым сервером осуществляется через обмен МАС-командами Rekeylnd/RekeyConf;

- устройство вычисляет F*NwkSlntKey & SNwkSlntKey & NwkSEncKey* из *NwkKey*;

- устройство вычисляет *AppSKey* из *АррКеу*.

Когда бит OptNeg не установлен, то:

- устройство использует LoRaWAN 1.0;

- команда *Rekeylnd* не отправляется устройством;

- устройство вычисляет *FNwkSlntKey и AppSKey* из *NwkKey*;

- устройство устанавливает значения ключей *SNwkSlntKey* *и NwkSEncKey* эквивалентными *FNwkSlntKey*.

Четыре сеансовых ключа F*NwkSlntKey*, *SNwkSlntKey*, *NwkSEncKey* и *AppSKey* формируются следующим образом:

Если OptNeg не установлен, то сеансовые ключи рассчитываются из NwkKey следующим образом:

AppSKey = aesl28\_encrypt(NwkKey,0x02 | JoinNonce | NetID | DevNonce | pad16)

FNwkSlntKey = aesl28\_encrypt(NwkKey, 0x01 | JoinNonce | NetID | DevNonce | pad16)

SNwkSlntKey = NwkSEncKey = FNwkSlntKey

Функция pad16 добавляет нулевой байт таким образом, чтобы длина данных стала кратной 16.

Значение MIC для сообщения подтверждения присоединения рассчитывается следующим образом:

cmac = aesl28\_cmac(*Nwkey*, MHDR | JoinNonce | NetID | DevAddr | DLSettings | RxDelay |CFList | CFListType)

*MIC = cmac[0..3]*

Если OptNeg установлен, то AppSKey рассчитывается из АррКеу следующим образом:

AppSKey = aesl28\_encrypt(AppKey,0x02 | JoinNonce | JoinEUI | DevNonce | pad16)

Сетевые сеансовые ключи рассчитываются из NwkKey следующим образом:

FNwkSlntKey = aesl28\_encrypt(NwkKey, 0x01 | JoinNonce | JoinEUI | DevNonce | pad16)

SNwkSlntKey = aesl28\_enciypt(Nwl<Key, 0x03 | JoinNonce | JoinEUI | DevNonce |рad16)

NwkSEncKey = aesl28\_encrypt(NwkKey, 0x04 | JoinNonce | JoinEUI | DevNonce | pad16)

В этом случае MIC вычисляется следующим образом:

cmac = aesl28\_cmac(*JSIntKey*,

JoinReqType | JoinEUI | DevNonce | MHDR | JoinNonce | NetID | DevAddr | DLSettings | RxDelay | CFList | CFListType)

MIC = cmac[0..3]

JoinReqType представляет собой одно байтовое поле для кодирования типа запроса на присоединение (JoinRequest или RejoinRequest), инициировавшего ответ JoinAccept.

**Таблица 17 - Кодировка JoinReqType**

|  |  |
| --- | --- |
| **Тип JoinReqType** | **Значение JoinReqType** |
| JoinRequest | OxFF |
| Тип JoinReqType | Значение JoinReqType |
| RejoinRequest type 0 | 0x00 |
| RejoinRequest type 1 | 0x01 |
| RejoinRequest type 2 | 0x02 |

Ключ, используемый для шифрования сообщения с подтверждением присоединения к сети, является функцией от сообщения с запросом присоединения (JoinRequest или RejoinRequest), инициировавшего его.

**Таблица 18 -Значение ключа шифрования JoinAccept**

|  |  |
| --- | --- |
| **Тип запроса** | **Ключ шифрования JoinAccept** |
| JoinRequest | *NwkKey* |
| RejoinRequest type = 0 или 1 или 2 | *JSEncKey* |

Сообщение подтверждения присоединения шифруется следующим образом:

aesl28\_decrypt(NwkKey or JSEncKey, JoinNonce | NetID | DevAddr | DLSettings |

RxDelay | CFList | CFListType | MIC)

Длина сообщения 16 или 32 байта.

Примечания

1 Алгоритм AES в режиме ЕСВ (Electronic Code Book). Данное решение позволяет шифровывать и расшифровывать сообщения по одному алгоритму (без необходимости разработки отдельного алгоритма для расшифрования) на стороне конечного устройства.

2 Создание этих четырех сеансовых ключей позволяет отделить инфраструктуру сервера сети от сервера приложения таким образом, чтобы операторы связи не могли расшифровывать данные приложения. Поставщик приложения соглашается, что сетевой оператор будет брать плату за переданные от конечного устройства пакеты и сохраняет полный контроль над ключом AppSKey, используемым для защиты данных приложения.

3 Версия используемого устройством протокола (LoRaWANI.O или LoRaWAN1.1) регистрируется на серверной стороне одновременно с DevEUI, NwkKey и АррКеу устройства.

Поле RX1 DRoffset определяет смещение между скоростью передачи данных восходящего канала связи и скоростью передачи данных нисходящего канала связи, используемое для связи с конечным устройством в первом окне приема (RX1). По умолчанию смещение равно 0. Смещение используется, чтобы учесть максимальные ограничения по мощности для базовых станций в некоторых регионах и для балансировки ограничений восходящей и нисходящей линий связи.

Фактические отношения между восходящей и нисходящей скоростью передачи данных определяются в региональных параметрах (приведены в разделе 9).

Задержка *RxDelay* следует тому же соглашению, что и поле *Delay* в команде *RXTim ingSetupReq*.

Поля CFIist и CFIistType являются необязательными, но должны либо оба присутствовать, либо оба отсутствовать.

Если сообщение с подтверждением присоединения к сети (Join-accept) получено после передачи:

- запроса на присоединение (или запроса на повторное присоединение типа 0 или 1) и, если поле CFIist отсутствует, то устройство переходит на частотные каналы «по умолчанию». Если CFIist присутствует, то он переопределяет все ранее заданные каналы. Параметры МАС-уровня (RXdelayl, скорость передачи данных RX2, ...) должны быть сброшены в значения по умолчанию.

- запроса на повторное присоединение типа 2 и, если поле CFIist отсутствует, то устройство должно сохранить свои текущие частотные каналы без изменений. Если CFIist присутствует, то он переопределяет *все* определенные ранее каналы. Все остальные параметры МАС-уровня (за исключением счетчиков кадров, которые сбрасываются) остаются без изменения.

Во всех случаях после успешной обработки сообщения JoinAccept устройство передает МАС-команду *Rekeylnd*, пока не получит команду *RekeyConf*. Прием восходящей команды *Rekeylnd* используется сетевым сервером, как сигнал для перехода к новому сеансу.

**6.4.2.4 Сообщение с запросом на повторное присоединение (ReJoin- request)**

После активации устройство может периодически передавать запрос на повторное присоединение (Rejoin-request) (помимо обмена данными, определенного приложением). Это сообщение с запросом повторного присоединения дает возможность периодически, на стороне сервера, инициализировать новый сеанс для конечного устройства. С этой целью сеть (сетевой сервер) отвечает сообщением подтверждения присоединения (Join-Accept). Это может быть использовано для передачи (перемещения) устройства между двумя сетями или в исходной сети для замены ключей и/или изменения devAddr устройства.

Сетевой сервер может также использовать окна приема RX1/RX2 после запроса повторного присоединения для передачи нормальных подтвержденных или неподтвержденных нисходящих сообщений, дополнительно передавая МАС- команды. Эта возможность полезна для сброса параметров приема устройства в случае, если состояние МАС-уровня рассинхронизовалось между устройством и сетевым сервером.

***Пример -*** Данный механизм может быть использован для изменения скорости передачи данных окна RX2 и смещения скорости передачи данных окна RX1 для устройства, которое более не доступно в нисходящей линии при использовании текущей конфигурации нисходящей линии связи.

Процедура повторного присоединения всегда инициируется конечным устройством, путем отправки сообщения с запросом повторного присоединения.

Примечание - В любое время сетевой сервер обрабатывает запросы на повторное присоединение (типа 0,1 или 2) и генерирует сообщения с подтверждением присоединения к сети, он должен поддерживать как старый сеанс (ключи и счетчики, если они есть), так и новый, пока не получит первый успешный пакет из восходящей линии связи, используя новый сеанс, после чего старый сеанс следует удалить. Во всех случаях обработка сообщения с запросом на повторное присоединение сетевым сервером похожа на обработку стандартного сообщения с запросом на присоединение к сети, в котором сетевой сервер в начале обработки сообщения определяет, должен ли он передать его серверу присоединения (Join Server), для формирования подтверждения присоединения (Join-accept) в ответ.

Существует три типа сообщений с запросом на повторное присоединение, которые могут быть переданы конечным устройством и соответствуют трем различным целям. Первый байт сообщения с запросом на повторное присоединение называется «Тип повторного присоединения» (Rejoin Туре) и используется для кодирования типа запроса на повторное присоединение. В таблице 19 описано назначение каждого типа сообщения с запросом на повторное присоединение.

**Таблица 19-Типы сообщений с запросом на повторное присоединение**

|  |  |
| --- | --- |
| **Тип**  **запроса** | **Описание и назначение** |
| 0 | Содержит NetlD+DevEUI. Используется для закрытия соединения на уровне сеанса с устройством, включая все параметры радио (devAddr, ключи сеанса, счетчики кадров, параметры радио, ..). Такие сообщения могут направляться устройствами только на домашние сетевые сервера, не на сервер присоединения (JoinServer). MIC этого сообщения может быть проверен только при транзитном обслуживании или домашним сетевым сервером. |
| 1 | Содержит JoinEUI+DevEUI. В точности эквивалентно исходному запросу присоединения Join-Request, но может передаваться поверх обычного прикладного трафика без отключения устройства. Только получающий его сетевой сервер может перенаправить на соответствующий устройству сервер Join.. Используется для восстановления утерянного криптографического контекста сеанса (например, сетевой сервер потерял ключи сеанса и не может связать (ассоциировать) устройство с JoinServer). Только JoinServer способен проверить MIC этого сообщения. |
| 2 | Содержит NetlD+DevEUI. Используется для замены ключей устройства или изменения его devAddr (devAddr, сеансовые ключи, счетчики кадров). Параметры радио остаются неизменными. Эти сообщения могут направляться на домашний сетевой сервер только посещаемыми сетями (из роуминга). Не могут быть отправлены сервером присоединения Join конечного устройства. MIC этого сообщения может бьггь проверен только при переправке (транзитном обслуживании) или домашним сетевым сервером. |

**6.4.2.4.1 Сообщение с запросом на повторное присоединение типа 0 или 2**

Сообщение с запросом на повторное присоединение типа 0 или 2 содержит NetID (идентификатор домашней сети устройства), DevEUI конечного устройства и значение 16 битного счетчика (RJcountO), в соответствии с рисунком 67.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер (в байтах)** | 1 | 3 | 8 | 2 |
| **Rejoin Request** | Rejoin Type = 0 или 2 | NetID | DevEUI | RJcount0 |

**Рисунок 67 - Структура сообщения с запросом на повторное присоединение**

*RJcount0* - счетчик, значение которого увеличивается с каждым переданным запросом на повторное присоединение 0 или 2 типа. RJcountO инициализируется в 0 каждый раз, когда подтверждение присоединения успешно обработано конечным устройством. Для каждого конечного устройства, сетевой сервер должен отслеживать и хранить последнее значение RJcount0 (так называемый RJcount0\_last), использованное конечным устройством. Он игнорирует запросы на повторное присоединение если (RJcount0 <= RJcount0\_last).

RJcount0 никогда не повторяется (не должен использоваться в цикле при переполнении). Если RJcountO достигает 216-1 , устройство должно прекратить передачу запроса на повторное присоединение 0 или 2 типа. Устройство может вернуться к состоянию присоединения к сети.

Примечание - Данный механизм предотвращает атаки посредством отправки предварительно записанных сообщений с запросами на повторное присоединение.

Код целостности сообщения (*MIC*) для сообщения с запросом на повторное присоединение рассчитывается следующим образом:

cmac = aesl28\_cmac(SNwkSIntKey, MHDR | Rejoin Type | NetID | DevEUI |

RJcount0)

MIC = *стас*[0..3]

Сообщение с запросом на повторное присоединение не шифруется.

Рабочий цикл устройства (duty-cycle) при передаче запросов на повторное присоединение 0 или 2 типа всегда должна быть <0.1 %.

Примечания

1 Сообщение с запросом на повторное присоединение типа 0 предполагается передавать (по замыслу) от одного раза в час до одного раза в несколько дней, в зависимости от варианта использования устройства. Это сообщение также может быть передано MAC командой ForceRejoinReq. Это сообщение может использоваться для переподключения мобильного устройства к гостевой сети при роуминге. Также может быть использовано для замены ключей или изменения devAddr статического устройства. Мобильные устройства, как ожидается, перемещаясь между сетями должны передавать это сообщение чаще, чем статические устройства.

2 Сообщение с запросом на повторное присоединение типа 2 предназначено только для обеспечения замены ключей конечного устройства. Это сообщение может передаваться только MAC командой ForceRejoinReq.

**6.4.2.4.2 Сообщение с запросом на повторное присоединение типа 1**

Аналогично запросу на присоединение сообщение с запросом на повторное присоединение типа 1 содержит JoinEUI и DevEUI конечного устройства (рисунок 68). Поэтому сообщение с запросом на повторное присоединение типа 1 может быть направлено серверу присоединения конечного устройства (Join Server) любым сетевым сервером, принявшим его. Запрос на повторное присоединение типа 1 может использоваться для восстановления связи с конечным устройством в случае полной потери сетевого сервера. Рекомендуется передавать сообщение с запросом на повторное присоединение типа 1 не реже одного раза в месяц.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Размер (в байтах)** | 1 | 8 | 8 | 2 |
| **Rejoin Request** | Rejoin Type = 1 | JoinEUI | DevEUI | RJcountl |

**Рисунок 68 - Структура сообщения с запросом на повторное присоединение**

**типа 1**

RJcount1 для запроса на повторное присоединение типа 1 - это другой счетчик, не RJCountO (который используется для запроса на повторное присоединение типа 0).

*RJcount1* - счетчик, значение которого увеличивается с каждым переданным запросом на повторное присоединение типа 1. Для каждого конечного устройства, сервер присоединения (join server) отслеживает и хранит последнее значение *RJcountl* (так называемый RJcount1\_last), использованное конечным устройством. Он игнорирует запросы на повторное присоединение если (RJcountl <= RJcountl\_last).

RJcountl никогда не повторяется для выданного JoinEUI. Периодичность отправки запроса на повторное присоединение типа 1 должна быть такой, чтобы не могло произойти переполнение счетчика и повторное использование его значений в период жизни устройства с заданным значением JoinEUI.

Примечание - Данный механизм предотвращает атаки, посредством отправки предварительно записанных сообщений с запросами на повторное присоединение.

Код целостности сообщения (*MIC*) для сообщения с запросом на повторное присоединение типа 1 рассчитывается следующим образом:

*cmac= aesl28\_cmac(JSIntKey,MHDR \ RejoinType \ JoinEUI \ DevEUI \ RJcount1)*

*MIC = cmac [0..3]*

Сообщение с запросом на повторное присоединение типа 1 не шифруется.

Рабочий цикл устройства (duty-cycle) при передаче запросов на повторное присоединение типа 1 всегда должен быть <0.01%.

Примечание - Сообщение с запросом на повторное присоединение типа 1 предполагается передавать от одного раза в день до одного раза в неделю. Это сообщение используется только в случае полной потери сервером криптографического контекста уровня сеанса. Это событие очень маловероятно, поэтому повторное подключение устройства с периодичностью от 1 раза в день до 1 раза в неделю считается приемлемым.

**6.4.2.4.3 Передача запроса на повторное присоединение**

В таблице 20 приведены возможные условия для передачи сообщения каждого типа запроса на повторное присоединение.

**Таблица 20- Возможные условия для передачи сообщения каждого типа запроса на повторное присоединение**

|  |  |  |
| --- | --- | --- |
| **Тип запроса на повторное присоединение (RejoinReq)** | **Передается автономно и периодически конечным устройством** | **Передается с MAC командой ForceRejoinReq** |
| 0 | X | X |
| 1 | X |  |
| 2 |  | X |

Сообщение с запросом на повторное присоединение типов 0 и 1 должно передаваться по любому, определенному для процедуры присоединения, каналу (см. раздел 9) на соответствующих случайно переключаемых (выбираемых) частотах.

Запрос на повторное присоединение типа 2 должен передаваться по любому, включенному в настоящий момент, каналу на соответствующих случайно переключаемых (выбираемых) частотах.

Запросы на повторное присоединение типов 0 и 2 передаваемые с использованием команды *ForceRejoinReq* должны использовать скорость передачи данных, указанную в MAC команде.

Запросы на повторное присоединение типа 0 (передаются периодически и автономно конечным устройством (с максимальной периодичностью, установленной командой RejoinParamSetupReq)) и запросы на повторное присоединение типа 1 должны использовать:

- скорость передачи данных и выходную мощность передатчика, используемые в настоящий момент для передачи прикладных данных (payload), если включен ADR;

- любую разрешенную для каналов присоединения скорость передачи данных и мощность передатчика по умолчанию, если ADR отключена. В этом случае рекомендуется использовать различные скорости передачи данных.

**6.4.2.4.4 Обработка запроса на повторное присоединение**

Для всех 3 типов запроса на повторное присоединение сетевой сервер может реагировать:

- сообщением с подтверждением присоединения (как описано в 6.4.2.3), если он хочет изменить сетевой идентификатор устройства (роуминг или замена ключей). В этом случае RJcount(0 или 1) заменяет DevNonce в процедуре получения ключей;

- нормальным (обычным) нисходящим кадром (сообщением) дополнительно содержащим МАС-команды. Этот нисходящий кадр должен быть отправлен в тот же канал, с той же скоростью передачи данных и с той же задержкой, что была задана для сообщения с подтверждением подключения, которое он заменяет.

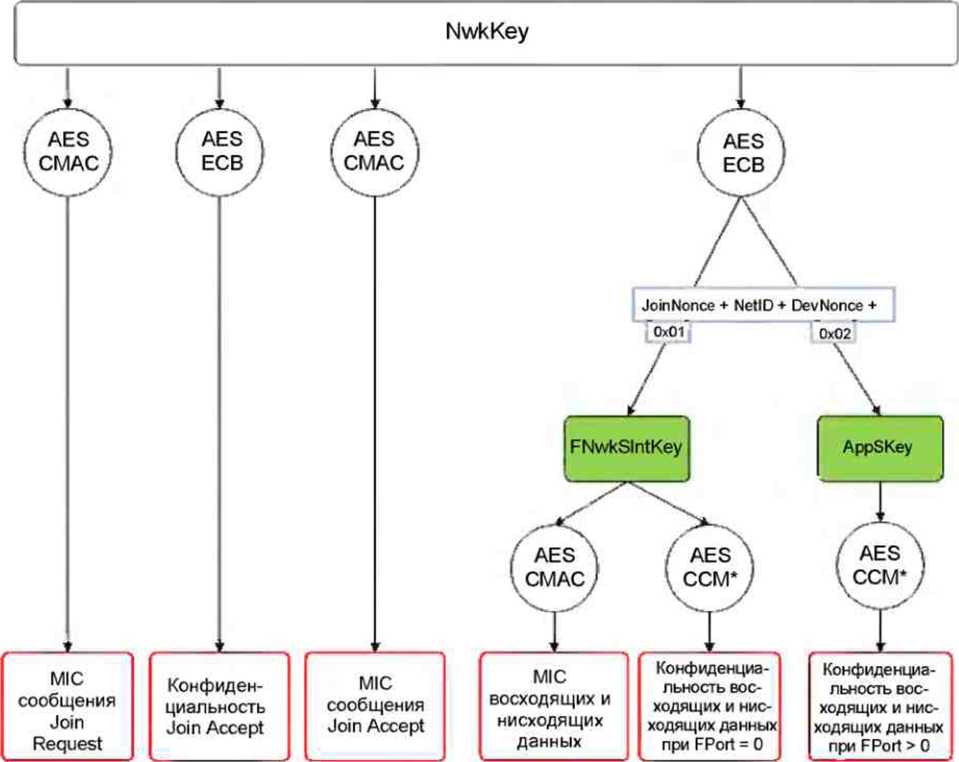
В большинстве случаев на попутные запросы на повторное присоединение типов 0 или 1 сеть не будет реагировать.

**6.4.2.5 Диаграмма формирования ключей**

На следующих диаграммах представлены схемы формирования ключей для случаев, когда устройство подключается к серверу сети LoRaWAN 1.0 или 1.1.

**Сервер сети LoRaWAN 1.0**

Когда устройство LoRaWAN 1.1 взаимодействует с сервером сети LoRaWAN 1.0.Х, все ключи вычисляются из первичного ключа *NwkKey*. Ключ устройства *АррКеу* не используется. Схема формирования ключа представлена на рисунке 69.

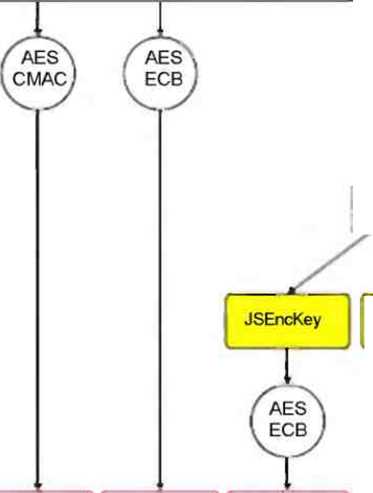
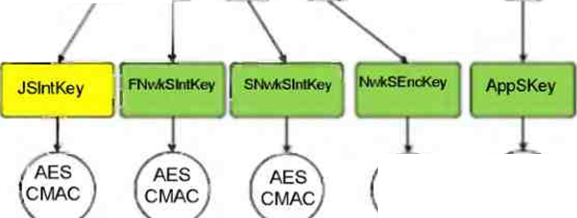


**Рисунок 69 - Схема формирования ключей, когда устройство подключается к**

**серверу сети LoRaWAN 1.0**

**Сервер сети LoRaWAN 1.1**

Схема формирования ключа представлена на рисунке 70.







**Рисунок 70 - Схема формирования ключей, когда устройство подключается к**

**серверу сети LoRaWAN 1.1**

**6.4.3 Активация через персонализацию (АВР)**

Активация через персонализацию напрямую связывает конечное устройство с конкретной сетью, минуя процедуру «запросприсоединения - подтверждение присоединения» (*join request - join accept*).

Активация конечного устройства через персонализацию означает, что *DevAddr* и четыре сеансовых ключа *FNwkSlntKey*, *SNwkSlntKey*, *NetSEncKey* и *AppSKey* хранятся непосредственно в устройстве, а не вычисляются из *DevEUI, JoinEUI* и *AppKey&NwkKey* во время процедуры присоединения. Устройство укомплектовано необходимой информацией для работы в конкретной сети на момент включения.

Каждое устройство должно иметь уникальный набор *FNwkSlntKey, SNwkSlntKey*, *NetSEncKey и AppSKey*. Компрометация ключей одного устройства не должна угрожать безопасности связи других устройств. Процесс создания (алгоритм вычисления) этих ключей должен быть такой, чтобы ключи не могли бьггь получены любым другим способом из общедоступной информации (например, адрес узла или devEUI конечного устройства).

Когда персонализированное конечное устройство подключается к сети в первый раз или после реинициализации, оно должно отправлять MAC команду Resetind в поле FOpt всех восходящих сообщений, пока не получит команду ResetConf из сети. После повторной инициализации (реинициализации) конечное устройство должно использовать свою конфигурацию по умолчанию (которая была использована, когда устройство было подключено к сети в первый раз).

**6.5 Задержка повторных передач**

Для восходящих кадров, для которых одновременно выполняются условия (1) и (3) или (2) и (3), существует ограничение на загрузку радиоэфира восходящими сообщениями. Условия, при которых действуют ограничения:

1) требующих подтверждения или ответа от сервера сети или сервера приложений;

2) являющихся повторной передачей по причине отсутствия ответа (подтверждения) от сервера;

3) объединенных внешним событием (отключение электричества, отключение сети, и т.д.), которое может инициировать одновременную синхронизацию большого количества устройств (>100), что может вызвать катастрофическую, тяжело восстанавливаемую ситуацию перегрузки радиосети.

Примечание - Примером такого восходящего кадра является JoinRequest, когда он выполняется группой конечных устройств решивших осуществить сброс МАС-уровня в случае сбоя сети. Вся группа конечных устройств начинает отправлять восходящие JoinRequest и прекращает только после получения от сети JoinResponse.

Для таких повторных отправок кадров, интервал между окончанием окна приема RX2 и следующей повторной передачей в восходящую линию связи должен быть случайным для каждого устройства (например, рекомендуется использование генератора псевдослучайных чисел с адресом устройства). Рекомендуется, чтобы рабочий цикл передачи таких сообщений соответствовал региональным параметрам (см п.9) и приведенным в таблице 21 ограничениям, в зависимости от того, чьи ограничения более строгие.

**Таблица 21 - Требования к допустимой загрузке радиоэфира**

|  |  |  |
| --- | --- | --- |
| **Описание периода** | **Момент передачи очередного кадра (t)** | **Допустимая загрузка радиоэфира** |
| В течение первого часа после включения питания или сброса | То < t < To+1h | Время передачи < 36 секунд за 1 час |
| После 1-го часа, в течение следующих 10 часов | To+1h < t < To+11h | Время передачи < 36 секунд за 11 часов |
| После первых 11 часов, в течение следующих 24 часов и за каждые последующие 24 часа | То+11+N < t < То+35+N  N≥0 | Время передачи < 8.7 секунд за 24 часа |

**7 Конечные устройства класса С**

**7.1 Режим связи конечного устройства**

Конечные устройства класса С используются там, где есть возможность использовать внешний источник питания (питаются от сети постоянного питания) и, следовательно, не требуется минимизировать время приема.

Устройства класса С не должны реализовывать возможности класса В.

Конечное устройство класса С большую часть времени прослушивает радиоэфир с параметрами окна приема RX2. Конечное устройство должно слушать в окне приема RX2, когда оно не передает (а), либо не принимает в окне приема RX1 (b), в соответствии с описанием на класса А. Для этого ему необходимо открыть маленькое (короткое) окно, использующее параметры RX2 между концом передачи в восходящую линию связи и началом окна приема RX1 и необходимо переключиться на параметры окна приема RX2, как только окно приема RX1 закроется, окно приема RX2 должно оставаться открытыми до тех пор, пока конечному устройству потребуется послать еще одно сообщение.

Примечания

1 Если устройство находится в процессе демодуляции нисходящего сообщения используя параметры RX2, в момент, когда должно бьггь открыто окно приема RX1, то устройство допжно прекратить демодупяцию и переключиться на прием в окне RX1.

2 Устройство класса С не может сообщить серверу, что оно поддерживает класс С. Сведения о принадлежности устройства к классу С должны попадать в сервер с прикладного уровня.

В случае если сообщение принимается устройством, работающим в режиме класса С, и требуется передача восходящего сообщения (нисходящая MAC команда- запрос или нисходящее сообщение, требующее подтверждения), устройство должно ответить в течение периода времени, известного как конечному устройству, так и сетевому серверу.

До истечения этого периода (тайм-аута), сеть не должна направлять какие-либо новые сообщения, требующие подтверждения или MAC команды на устройство. После истечения этого периода или после приема любого восходящего сообщения, сети разрешено посылать новое нисходящее сообщение.

**7.1.1 Длительность второго окна приема для класса С**

Устройства класса С реализуют те же два окна приема, что и устройства класса А, но они не закрывают окно приема RX2 до момента отправки очередного восходящего сообщения (рисунок 75). Поэтому они могут получать нисходящие сообщения в окне приема RX2 почти в любое время, в том числе нисходящие сообщения, отправленные с целью передачи MAC команды или подтверждения получения сообщения (АСК). Короткое окно прослушивания на частоте и скорости передачи данных RX2, так же открывается между окончанием передачи и началом приема в окне RX1.

Рисунок 75 - Временной график приема сообщений для класса С

**7.1.2 Многоадресная рассылка для класса С**

Аналогично классу В, устройства класса С могут принимать многоадресные нисходящие пакеты. Адрес многоадресной рассылки и соответствующие сетевой сеансовый ключ и сеансовый ключ приложения должны приходить на уровне приложения.

Примечание - Многоадресная рассылка может использоваться для многоадресной передачи следующих данных: обновление встроенного программного обеспечения, единое время, альманах и эфемериды GPS/GLONASS -спутников (для ускоренного определения координат конечными устройствами) и т.д.

Ограничения, распространяющиеся на многоадресные нисходящие сообщения для класса С:

- сообщения передаются только в нисходящем канале связи;

- сообщения не должны нести MAC команды, ни в области *FOpts*, ни в поле данных FRMPayload на порт 0;

- биты *АСК* и *ADRACKReq* должны быть равны 0;

- поле *МТуре* должно нести значение, соответствующее нисходящему сообщению, не требующему подтверждения (МТуре = Unconfirmed Data Down);

- бит *FPending* - должен указывать на то, что имеются еще многоадресные данные для отправки.

Примечание - Учитывая, что устройство класса С сохраняет активным свой приемник большую часть времени, то бит FPending не вызывает какого-либо конкретного поведения конечного устройства.

**7.2 МАС-команды**

Все команды, описанные для класса А, должны быть реализованы в устройствах класса С. Для устройств класса С дополнительно определены MAC команды, указанные в таблице 22.

**Таблица 22 - МАС-команды для устройств класса С**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CID** | **Команда** | **Передается** | | **Краткое описание** |
| **КУ** | **БС** |
| 0x20 | *DeviceModelnd* | х |  | Используется конечным устройством для обозначения его текущего режима работы (класс А или С). |
| 0x20 | *DeviceModeConf* |  | X | Используется сетью для подтверждения команды DeviceModelnd. |

**7.2.1 Режим работы устройства (DeviceModelnd, DeviceModeConf)**

С помощью команды *DeviceModelnd* конечное устройство извещает сеть о режиме своей работы в классе А или классе С. Команда имеет данные размером один байт, согласно рисунку 76.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **DeviceModelnd Payload** | Класс (Class) |

**Рисунок 76 - Атрибут команды DeviceModelnd**

Значения классов для команды DeviceModelnd представлены в таблице 23.

**Таблица 23- Значения классов для команды DeviceModelnd**

|  |  |
| --- | --- |
| **Поле «Класс» (Class)** | **Значение** |
| Класс A (Class А) | 0x00 |
| RFU | 0x01 |
| Класс С (Class С) | 0x02 |

Когда сетевой сервер получает команду *DeviceModelnd*, он отвечает на нее командой *DeviceModeConf*. Устройство должно включать команду *DeviceModelnd* во все восходящие сообщения, пока не получит команду *DeviceModeConf*.

Устройство должно переключить режим работы, как только первая команда *DeviceModelnd* будет передана.

Примечание - Для устройств с батарейным питанием рекомендуется при переходе от класса А к классу С реализовать механизм тайм-аутов на прикладном уровне, чтобы гарантировать, что устройство не задержится на неопределенный срок в режиме класса С при отсутствии связи с сетью.

Команда *DeviceModeConf* содержит один байт данных как в рисунке 77.

|  |  |
| --- | --- |
| **Размер (в байтах)** | 1 |
| **DeviceModeConf Payload** | Class |

**Рисунок 77 - Структура команды DeviceModeConf**

Параметр Class определяется также, как для MAC команды *DeviceModelnd.*

**8 Примеры реализации**

Ниже приведены примеры, иллюстрирующие применение настоящего стандарта.

**8.1 Диаграмма передачи восходящего сообщения с подтверждением**

Следующая диаграмма иллюстрирует шаги, выполняемые конечным устройством, которое пытается передать два восходящих сообщения (DataO и Datal) с требованием подтверждения. Параметр NbTrans этого устройства должен быть больше или равен 2, чтобы этот пример был действительным (т.к. первый подтвержденный кадр передается дважды).

**Рисунок 78 - Диаграмма передачи восходящего сообщения с подтверждением**

Конечное устройство сначала передает кадр данных, требующий подтверждения, содержащий данные DataO, в произвольный момент времени и на произвольном канале. Значение счетчика кадров (Си) формируется путем добавления 1 к предыдущему счетчику кадров восходящей линии связи. Сервер сети принимает кадр и генерирует кадр в нисходящую линию связи. В нисходящем сообщении установлен бит АСК т.е. подтверждение получения предыдущего сообщения. Нисходящее сообщение передается с задержкой RECEIVE\_DELAY в первое окно приема RX1конечного устройства. Данное нисходящее сообщение использует ту же скорость передачи данных и тот же частотный канал, что и предыдущее восходящее сообщение с DataO. Счетчик кадров в нисходящей линии связи (Cd) получается путем добавления 1 к предыдущему значению счетчика кадров в нисходящей линии для данного экземпляра конечного устройства. Если в сервере нет данных, ожидающих передачи в конечное устройство, то сеть должна генерировать сообщение без прикладных данных. В данном примере кадр, несущий бит АСК, не принимается конечным устройством из-за помех в радиоканале.

Если конечное устройство не получает в течении времени ACK\_TIMEOUT ни в одном из окон приема (RX1 или RX2) кадр с битом АСК, то конечное устройство может повторно отправить те же данные (DataO) с тем же счетчиком кадров (Си). Эта повторная отправка должна выполняться на другом частотном канале и должна соответствовать ограничению рабочего цикла (DutyCycle), как и любая другая передача в восходящем канале. Если на этот раз конечное устройство принимает в нисходящем канале подтверждение (бит АСК) во время своего первого окна приема RX1, то конечное устройство затем может передавать следующий кадр (Datal) на новый канал.

Кадр нисходящей линии связи может нести комбинацию сведений: подтверждение предыдущего сообщения (АСК), МАС-команды и прикладные данные.

**8.2 Диаграмма передачи нисходящего сообщения с подтверждением**

Следующая диаграмма иллюстрирует типовую последовательность передачи сообщения с подтверждением в нисходящую линию связи.

**Рисунок 79 - Диаграмма передачи нисходящего сообщения с подтверждением**

Обмен данными инициируется конечным устройством класса А, передающим сообщение (Data), не требующее подтверждения. Сервер сети использует первое окно приема (RX1) в нисходящей линии связи для передачи сообщения, требующего подтверждения в направлении конечного устройства. Передача нисходящего сообщения осуществляется на канале предыдущего восходящего сообщения. Конечное устройство, после приема данного нисходящего сообщения, передает сообщение с битом АСК, подтверждающим получение предыдущего сообщения. Данное восходящее сообщение может содержать данные или МАС-команды, и передается на новом канале, выбранным случайным образом.

Примечание - Чтобы конечные устройства были максимально простыми и имели как можно меньше состояний, они могут передавать пустое сообщение с подтверждением (без прикладных данных) сразу после приема нисходящего сообщения, требующего подтверждения. В качестве альтернативы, конечное устройство может отложить передачу подтверждения, чтобы передать его вместе со своими следующими прикладными данными.

**8.3 Диаграмма передачи очереди нисходящих сообщений**

Следующая диаграмма иллюстрирует управление очередью нисходящих сообщений с помощью бита FPending.

Бит FPending может быть установлен только в кадре нисходящей линии связи и информирует конечное устройство о том, что сервер имеет несколько сообщений в очереди для передачи данному конечному устройству. В восходящих сообщениях бит FPending игнорируется сервером сети.

Если кадр с установленным битом FPending=1 требует подтверждения, то конечное устройство должно сделать это, как описано выше (п.8.2). Если подтверждение не требуется, конечное устройство может отправить пустое сообщение (без прикладных данных), чтобы открыть очередные окна приема (RX1 и RX2) или дождаться, когда в конечном устройстве появятся прикладные данные, которые необходимо передать в сервер сети.

Примечание - Бит FPending не зависит от подтверждения (АСК).

Пример 1 приведен на рисунке 80.

**Рисунок 80 - Диаграмма передачи очереди нисходящих сообщений (пример 1)**

В данном примере сеть передает в конечное устройство два сообщения, требующих подтверждения.

Обмен сообщениями инициируется конечным устройством класса А посредством передачи сообщения в восходящую линию связи. Сообщение передается на частотном канале chA. Сеть использует первое окно приема (RX1) для передачи на канале chA данных Data0 с установленными битом FPending и требованием подтверждения. Устройство, получив сообщение с битом FPending=1, передает на новом частотном канале chB подтверждение приема данного сообщения, передавая обратно пустой кадр с битом АСК.

С задержкой в RECEIVE\_DELAY1 секунд, сеть на канале chB передает в устройство второе сообщение (Datal), требуя подтвердить получение сообщения, но с бит FPending теперь равным 0.

Конечное устройство подтверждает получение (Datal) на канале chC.

Пример 2 приведен на Рисунке 81.

**Рисунок 81 - Диаграмма передачи очереди нисходящих сообщений (пример 2)**

В данном примере сообщения в нисходящей линии связи являются «сообщениями, не требующими подтверждения» конечным устройством.

Конечное устройство при получении сообщения Data0, не требующего подтверждения, но с установленным битом FPending - отправляет в сеть пустое сообщение без прикладных данных. Это первое восходящее сообщение не принимается сетью по причине помех. Если ни одно нисходящее сообщение не было получено в течение двух последующих окон приема (RX1 и RX2), то конечное устройство должно повторить передачу пустого восходящего сообщения. После получения сервером сети пустого сообщения в одно из окон приема (RX1 и RX2) отправляется следующее нисходящее сообщение (cd+1) из очереди.

Примечание - Подтверждение никогда не отправляется дважды.

Пример 3 приведен на рисунке 82.

Бит FPending, бит АСК и прикладные данные могут одновременно присутствовать в одном нисходящем сообщении.

**Рисунок 82 - Диаграмма передачи очереди нисходящих сообщений (пример 3)**

Конечное устройство отправляет в восходящую линию связи данные, требующие подтверждения. Сервер сети может ответить сообщением, содержащим: подтверждение получения сообщения из восходящей линии связи, данные нисходящей линии связи Data, требующие подтверждения и поле FPending=1, информирующее о наличии очереди сообщений для данного устройства.

**9 Региональные параметры**

**9.1 KZ 865-868**

**9.1.1 Формат поля «Преамбула»**

**Таблица 24-Структура поля «Преамбула»**

|  |  |  |
| --- | --- | --- |
| **Модуляция** | **Синхрослово** | **Размер поля «Преамбула»** |
| LoRa | 0x34 | 8 символов |
| GFSK | 0хС194С1 | 5 байт |

**9.1.2 Частотные каналы**

Используемые каналы связи должны соответствовать требованиям регламентирующих документов Государственной комиссии по радиочастотам.

Все доступные частотные каналы могут использоваться оператором связи по его усмотрению. Три канала «по умолчанию» должны быть реализованы в каждом конечном устройстве (таблица 25). Данные каналы являются обязательными и не могут быть отредактированы МАС-командой NewChannelReq. Эти каналы являются минимальным набором, который должен всегда прослушиваться всеми радио-шлюзами сети связи.

**Таблица 25-Каналы «поумолчанию»**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер**  **канала** | **Модуляция** | **Частота**  **канала,**  **МГц** | **Полоса**  **частот,**  **кГц** | **Скорость** | | **Рабочий**  **цикл**  **(DutyCycle)** | **Мощность,**  **мВт/дБм** |
| 1 | LoRa | 865,1 | 125 | DRO… | DR5 | <10% | 25/14 |
| 2 | LoRa | 865,3 | 125 | DRО… | DR5 | <10% | 25/14 |
| 3 | LoRa | 865,5 | 125 | DRO… | DR5 | <10% | 25/14 |

**9.1.3 Кодирование скорости и мощности**

Кодирование скорости передачи данных (DR) определено в таблице 28.

**Таблица 28- Кодировка скорости передачи данных**

|  |  |  |
| --- | --- | --- |
| **Скорость**  **(DataRate)** | **Конфигурация** | **Физическая скорость передачи данных, бит/с** |
| 0 | LoRa: SF12/125 кГц | 250 |
| 1 | LoRa: SF11 /125 кГц | 440 |
| 2 | LoRa. SF10 /125 кГц | 980 |
| 3 | LoRa: SF9/125 кГц | 1760 |
| 4 | LoRa: SF8/125 кГц | 3125 |
| 5 | LoRa: SF7/125 кГц | 5470 |
| 6 | LoRa: SF7 / 250 кГц | 11000 |
| 7 | FSK: 50 кбит/с | 50000 |
| 8.14 | Зарезервировано | Зарезервировано |
| 15 | Сохранить предыдущее значение | |

Кодирование выходной мощности конечного устройства (TXPower) представлено в таблице 29.

**Таблица 29- Кодирование выходной мощности конечного устройства**

|  |  |
| --- | --- |
| **TXPower** | **Физическое значение** |
| 0 | 27 дБм (Зарезервировано) |
| 1 | 20 дБм (Зарезервировано) |
| 2 | 16 дБм (Зарезервировано) |
| 3 | 14 дБм |
| 4 | 12 дБм |
| 5 | 10 дБм |
| 6 | 8 дБм |
| 7 | 6 дБм |
| 8 | 4 дБм |
| 9 | 2 дБм |
| От 10 до 14 | Зарезервировано |
| 15 | Сохранить предыдущее значение |

Значения «Сохранить предыдущее значение» - используется в МАС-команде LinkADRReq при редактировании диапазона допустимых скоростей и мощности передачи пакета.

**9.1.4 Частотные каналы, передаваемые в CFList в подтверждении присоединения (JoinAccept)**

Поле CFList (рисунок 83) представляет собой список из пяти частотных каналов. Каждая частота кодируется как целое число без знака 24 бита (три байта). Все эти каналы могут использоваться со скоростью от DR0 до DR5 в полосе 125 кГц. За списком частот следует один резервный байт (RFU).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Размер (байт)** | 3 | 3 | 3 | 3 | 3 | 1 |
| **CFList** | Freq ch3 | Freq ch4 | Freq ch5 | Freq ch6 | Freq ch7 | RFU |

**Рисунок 83 - Структура поля CFList**

Фактическая частота канала в Гц равна 100\*[Freq chX], Это позволяет установить частоту канала в диапазоне от 100 МГц до 1,67 ГГц с шагом 100 Гц.

Неиспользуемые каналы имеют значение частоты равное 0.

Поле *CFList* является необязательным в сообщении «Подтверждение присоединения», и его присутствие может быть выявлено по длине сообщения «Подтверждение присоединения». Если поле *CFList* присутствует, то устройство заменяет все предыдущие каналы, хранящиеся в конечном устройстве, кроме двух каналов «по умолчанию». Новые каналы сразу могут быть использованы конечным устройством.

**9.1.5 Маска каналов в МАС-команде LinkAdrReq**

Когда поле *ChMaskCntl* равно 0, то поле *ChMask* индивидуально включает/выключает каждый из от 1 до 16 каналов.

**Таблица 30-Маска каналов в МАС-команде**

|  |  |
| --- | --- |
| **ChMaskCntl** | **Область применения попя ChMask** |
| 0 | Каналы от 1 до 16 |
| 1 | Зарезервировано |
| ... | ... |
| 4 | Зарезервировано |
| 5 | Зарезервировано |
| 6 | Все каналы включены.  Устройство должно включить все известные каналы, независимо от значения поля ChMask |
| 7 | Зарезервировано |

Если значение поля ChMaskCntl соответствует «Зарезервировано», то конечное устройство должно отклонить МАС-команду и отключить бит «АСК» в ответе.

**9.1.6 Максимальный размер поля данных (MACPayload)**

Максимальный размер поля данных (М) приведен в таблице ниже. Он получен из ограничений физического уровня, в зависимости от эффективной скорости модуляции. Максимальная длина прикладных данных (FRMPayload) в отсутствие дополнительного поля управления FOpt (Л/) предоставляется только для информации. Значение N может быть соответственно меньше, если поле FOpt содержит данные длинною от 1 до 15 байт.

**Таблица 31- Максимальный размер поля данных**

|  |  |  |
| --- | --- | --- |
| **DataRate** | ***М*** | ***N*** |
| 0 | 59 | 51 |
| 1 | 59 | 51 |
| 2 | 59 | 51 |
| 3 | 123 | 115 |
| 4 | 230 | 222 |
| 5 | 230 | 222 |
| 6 | 230 | 222 |
| 7 | 230 | 222 |
| От 8 до 15 | Не определено | Не определено |

**9.1.7 Окна приема RX1/RX2**

В окне приема RX1 должен использоваться тот же частотный канал, который использовался при передаче предыдущего восходящего сообщения.

Скорость передачи данных в окне RX1 зависит от скорости передачи данных по восходящей линии связи и значения RX1 DROffset, согласно таблице 32.

**Таблица 32 - Скорость передачи данных в окне RX1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Скорость в восходящем канале** | **Скорость в приемном окне RX1** | | | | | |
| **В зависимости от значения RX1 DROffset** | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** |
| DR0 | DRO | DRO | DRO | DRO | DRO | DRO |
| DR1 | DR1 | DRO | DRO | DRO | DRO | DRO |
| DR2 | DR2 | DR1 | DRO | DRO | DRO | DRO |
| DR3 | DR3 | DR2 | DR1 | DRO | DRO | DRO |
| DR4 | DR4 | DR3 | DR2 | DR1 | DRO | DRO |
| DR5 | DR5 | DR4 | DR3 | DR2 | DR1 | DRO |

Допустимые значения для RXIDROffset находятся в диапазоне от 0 до 5. Значения в диапазоне от 6 до 7 зарезервированы для будущего использования.

В приемном окне RX2 используется фиксированная частота и скорость передачи данных. Параметры по умолчанию: 867,5 МГц на скорости DR2 (SF12, 125 кГц).

**9.1.8 Настройки по умолчанию**

Параметры, указанные в таблице 33, рекомендуется использовать по умолчанию.

**Таблица 33- Настройки по умолчанию**

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| RECEIVE\_DELAY1 | 1 сек |
| RECEIVE\_DELAY2 | RECEIVE\_DELAY1 + 1 |
| JOIN ACCEPT DELAY1 | 5 сек |
| JOIN ACCEPT DELАY2 | 6 сек |
| MAX\_FCNT\_GAP | 16384 |
| ADR\_ACK\_LIMIT | 64 |
| ADR\_ACK\_DELAY | 32 |
| ACK\_TIMEOUT | 1 ...3 сек (случайное значение в интервале) |

Если фактические значения параметров, реализованные в конечном устройстве, отличаются от значений по умолчанию (например, конечное устройство использует более короткую задержку RECEIVE\_DELAY1 и RECEIVE\_DELAY2), то эти параметры должны быть переданы серверу сети во время ввода в эксплуатацию конечного устройства. Сервер сети может не принимать значения параметров, отличных указанных по умолчанию.

**Библиография**

[1] Правил присвоения полос частот, радиочастот (радиочастотных каналов), эксплуатации радиоэлектронных средств и высокочастотных устройств, а также проведения расчета электромагнитной совместимости радиоэлектронных средств гражданского назначения, утвержденные приказом и.о. Министра по инвестициям и развитию Республики Казахстан от 21 января 2015 года № 34 *(с*[*изменениями и дополнениями*](https://online.zakon.kz/document/?doc_id=33312550)*по состоянию на 22.10.2020 г.)*

[2] Таблицы распределения полос частот между радиослужбами Республики Казахстан в диапазоне частот от 3 кГц до 400 ГГц для радиоэлектронных средств всех назначений утвержденные приказом и.о. Министра по инвестициям и развитию Республики Казахстан от 20 января 2015 года № 22

[3] IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-2699 Rate Wireless Personal Area Networks (LR-WPANs), IEEE Std 802.15.4TM-2011 (Revision 2700 of IEEE Std 802.15.4-2006), September 2011.

[4] The AES-CMAC Algorithm, June 2006.

[5] LoRaWAN Regional parameters v1.1, LoRa Alliance

[6] LoRaWAN backend Interfaces specification v1.0, LoRa Alliance

**УДК 004.783 МКС 35.020.35.110**

**Ключевые слова:** Интернет вещей, информационные технологии, LoRaWAN, беспроводная передача данных

|  |  |  |
| --- | --- | --- |
| Разработчик: | АО «Информационно-аналитический центр нефти и газа» | |
| Генеральный директор | | \_\_\_\_\_\_\_\_ | |
| Директор Департамента | | \_\_\_\_\_\_\_\_\_\_ | |