TR007 Developing LoRaWAN® Devices **1.0**
Technical Recommendations

Copyright © 2021 LoRa Alliance®, Inc. All rights reserved.

TR007 Developing LoRaWAN ® Devices

Technical Recommendations

Authored by the LoRa Alliance Technical Committee

Technical Committee Chair and Vice-Chair:
A.YEGIN (Actility)**,** O.SELLER (Semtech)

Editor:
D.KJENDAL (Senet)

Contributors:
S.BALL (Senet), J.COUPIGNY (ST), R.FULLER (Semtech), S.JILLINGS (Semtech), J.KNAPP (Semtech), S.LEE (Semtech), F.LU (Semtech), S.NAIK (Semtech), B.PARATTE (Semtech), H.RITTER (Minol), S.SHEN (Semtech), J.STOKKING (The Things Network), B.VERSTEEG (DISH), A.YEGIN (Actility), X.YU (Alibaba Group), O.SELLER (Semtech), D.KJENDAL (Senet)

Version: 1.0
Date: February 3, 2021
Status: Final

Contents

Contents 3
Tables 5
1 Introduction 6
1.1 Conventions 6
2 Provisioning 7
2.1 Introduction 7
2.2 Extended-Unique-Identifier (EUI) 7
2.3 Device Address (DevAddr) 7
2.4 Security Keys 8
2.5 OTAA Device Provisioning 8
2.5.1 Device EUI (DevEUI) 8
2.5.2 JoinEUI 8
2.5.3 Root Key(s) 9
2.6 ABP Devices 10
2.6.1 DevAddr 10
2.6.2 Session Keys 10
2.6.3 Physical Provisioning 11
3 LoRaWAN Protocol Stack 12
3.1 Join Procedure 12
3.1.1 Description 12
3.1.2 Recommended Practice 12
3.2 Fixed Channel Plan Join Process Optimization 12
3.2.1 Description 12
3.2.2 Recommended Practice 12
3.3 Detecting Loss of Network Connectivity 13
3.3.1 Description 13
3.3.2 Recommended Practice 13
3.4 Reacting to Loss of Network Connectivity 14
3.4.1 Description 14
3.4.2 Recommended Practice 14
3.5 Rolling Session Keys 15
3.5.1 Description 15

3.5.2 Recommended Practice 15
3.6 DevNonce Values SHALL Not Be Reused 15
3.6.1 Description 15
3.6.2 Recommended Practice 16
3.7 Avoiding Synchronous Behavior 16
3.7.1 Description 16
3.7.2 Recommended Practice 16
3.8 Avoiding Congestion Collapse 16
3.8.1 Description 16
3.8.2 Recommended Practice 17
3.9 Discontinuing Retransmissions 18
3.9.1 Description 18
3.9.2 Recommended Practice 18
3.10 Default Channels 19
3.10.1 Description 19
3.10.2 Recommended Practice 19
3.11 FcntUp/FcntDown SHALL Use 32 Bits 19
3.11.1 Description 19
3.11.2 Recommended practice 19
3.12 Frame Counters Incremented only with Frame Transmission 20
3.12.1 Description 20
3.12.2 Recommended Practice 20
3.13 OTAA Required Persistent Values 20
3.13.1 Description 20
3.13.2 Recommended Practice 20
3.14 ABP Required Persistent Values 21
3.14.1 Description 21
3.14.2 Recommended Practice 21
3.15 ADR (Adaptive Data Rate) Support 21
3.15.1 Description 21
3.15.2 Recommended Practice 22
3.16 ADR Back Off 22
3.16.1 Description 22

3.16.2 Recommended Practice 22
3.17 Duty Cycle Limitations 22
3.17.1 Description 22
3.17.2 Recommended Practice 22
3.18 Transmit Power 23
3.18.1 Description 23
3.18.2 Recommended Practice 23
3.19 Class B/C Cell Update 23
3.19.1 Description 23
3.19.2 Recommended practice 23
4 Glossary 25
5 NOTICE OF USE AND DISCLOSURE 26

Tables

Table 1: Transmit duty-cycle limitations18

1 Introduction

This technical recommendation is intended to provide guidance to end-device and LoRaWAN® protocol stack developers to help ensure they produce well-behaved and interoperable products. The document is structured in a series of numbered sections. While this document does not constitute a new technical specification, normative language is used in each section so that behavior compliant with the recommendation is clear. Further, this will allow the implementor to claim compliance with a specific section of the recommendation.

1.1 Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The tables in this document are normative. The figures in this document are informative. The notes in this document are informative.

2 Provisioning

2.1 Introduction

LoRaWAN® end-devices, both Over-the-Air-Activation (OTAA) and Activation-By-Personalization (ABP), must be properly provisioned for operation on a LoRaWAN® compliant network. OTAA devices are provisioned with a DevEUI, JoinEUI and one or more root keys. ABP devices are provisioned with a DevEUI, DevAddr and one-time session keys.

All end-devices are identified through the DevEUI (device-EUI). All end-devices are also provisioned with a shared secret (or secrets) which are used to authenticate and encrypt frames to and from the network.

2.2 Extended-Unique-Identifier (EUI)

LoRaWAN® uses EUI's to identify various elements of the network architecture. End-devices are identified by a DevEUI (device EUI) and Join Servers are identified by a JoinEUI. The EUI's in LoRaWAN® specifications are always EUI-64 (64-bit EUI's). Other architectural elements may use EUI's as identifiers in future.

An EUI is a globally unique identifier, allocated from a pool of EUI's owned by the organization which has been allocated them by the IEEE. Organizationally-Unique-Identifier (OUI) form the most significant 24 bits of the EUI for MA-L address blocks. MA-M and MA-S address blocks are based on a 36-bit OUI-36. The IEEE is the Registration Authority which assigns the OUI or OUI-36 and the relevant MA-L, MA-M or MA-S address blocks to an organization. EUI's SHALL be globally unique.

Vendors must purchase an OUI from the IEEE and then allocate their DevEUI and JoinEUI addresses from the range of addresses that the OUI defines. Vendors can choose whether to keep their company name and address confidential (for an added, annual fee).

The consequence of a vendor incorrectly allocating DevEUIs or JoinEUIs (either by not owning an OUI block or improperly allocating it) is that they may collide with EUIs created by the true owner of the OUI block. This can lead to one or both devices failing to properly operate on LoRaWAN networks (this is true for both public and private networks). For this reason, implementors SHALL refrain from using OUIs to which they are not entitled to (including random OUIs or other vendors' OUIs) when creating DevEUIs and JoinEUIs.

2.3 Device Address (DevAddr)

The DevAddr is a short-form address used to aid in the identification of an end-device. It is allocated by the Network Operator and consists of Network-ID and Host-ID (in the same manner as IP addresses). The DevAddr is 32-bits wide and is not guaranteed to be unique. The NetID is allocated by the LoRa Alliance® to member companies which request one. If the operator does not have a NetID allocated by the LoRa Alliance® they SHALL use NetID 0 or 1, the private NetIDs.

If the network operator does not follow this guideline and uses another network operator's NetID when allocating DevAddrs, there are two consequences: first, it is impossible to efficiently filter traffic at the gateway, creating additional backhaul usage and requiring filtering at the network server itself; second, uplink frames of the misidentified device received by another network operator will not be forwarded to the real home network of the device, instead they will be forwarded to the network of the real owner of the NetID; this prevents the device from ever being able to take advantage of roaming network coverage.

2.4 Security Keys

In LoRaWAN® specifications, all security keys are 128-bits wide. Root keys are used only to derive session keys and in the JoinRequest and JoinAccept messages. Session keys are used to form Message Integrity Codes (MICs) and to encrypt and decrypt all messages other than the JoinRequest and JoinAccept messages.

2.5 OTAA Device Provisioning

OTAA devices must join a network to gain connectivity. The join process consists of a JoinRequest issued by the end-device, followed by a JoinAccept from the network (granting access and providing a DevAddr and session key derivation material) and finally confirmed by the first uplink from the end-device using the new session keys. This section will describe what must be provisioned on the end-device and network to support OTAA operation.

2.5.1 Device EUI (DevEUI)

The DevEUI is a globally unique EUI-64, legitimately allocated to the end-device by the organization owning the OUI. When allocating a DevEUI, consider that EUI's are (relatively) scarce resources which cannot be re-used. Allocate them efficiently and with safeguards in the manufacturing process to guarantee their uniqueness.

Do not make-up or invent EUI's for use. EUI's of all (binary) ones or zeros are invalid and SHALL NOT be used. Radio modules almost always are provisioned by the manufacturer with a valid DevEUI. Device makers implementing chip-down designs or radio modules without a valid DevEUI SHALL acquire and allocate a valid, globally unique DevEUI from a range they have been granted the right to administer.

2.5.2 JoinEUI

The JoinEUI (also known as AppEUI in earlier versions of the specifications) is the EUI-64 address of the Join Server which has been provisioned with the root key material paired with the end-device. The Join Server must support (at minimum) the configuration of DevEUI/JoinEUI pairs with the device's associated root key(s). The JoinServer is responsible for validating the end-device's JoinRequest, its DevNonce and the next JoinNonce. This JoinEUI is used by the end-device to address the JoinRequest message.

Do not make-up or invent EUI's for use. Do not assign a different JoinEUI to every device, as this defeats the intention of the JoinEUI: to provide a hierarchical address resolution to the Join Server. The JoinEUI SHALL be a valid, globally unique EUI allocated from a range the Join Server operator has been granted the right to administer. Many devices are expected to use the same JoinEUI (and thus the same Join Server).

2.5.3 Root Key(s)

2.5.3.1 Root Key Generation

Root Keys must be generated in such a fashion that there is no derivable pattern between the root keys of multiple devices.

By ensuring a high degree of entropy (randomness) between the keys of a set of devices the manufacturer ensures that the compromise of a single device does not lead to the compromise of additional devices. That keys conform to this model is critical, as it makes the economics of cracking individual devices infeasible for large-scale attacks.

Two techniques are often used for Root Key generation. The first is to create a large set of unique keys, each with its own input entropy (randomness). The second is to create a master root key for a set of devices and then use one-way, keyed, cryptographic hash function to derive the unique root keys for each individual device. Creating individual, unique keys with a high degree of entropy removes the weakness of an attacker obtaining the master root key, but it is often computationally expensive and more difficult to securely manage in production. As a result, many implementations follow the master root key model which minimizes the number of secrets which must be individually managed. It is preferable that a Hardware Security Module, configured with the master key, be used to securely generate root keys on demand. The master root key must be kept secure, as compromise of this key may compromise the entire set of devices provisioned with keys derived from it. The master root key SHALL NOT be kept on an end device, as this would raise the value of the attack to the point where it becomes economically feasible.

It is not acceptable to use the same root key on multiple devices. Additionally, root keys SHALL NOT be reversibly based on DevEUI or any other easily guessed scheme. Keys SHALL NOT be all 0’s or all 1’s or any other simple pattern. Obtain security keys using a state-of-the-art cryptographic process that allows minimal transport of keys in plain text.

2.5.3.2 Root Key Delivery and Storage

The various parties (manufacturer, distributor, integrator, operator, etc.) SHALL strictly limit access to root key material.

An attacker is likely to try to compromise the system at its weakest point. The LoRaWAN® protocol describes mechanisms that use security keys to secure traffic between the various elements of the system. This design has been subjected to extensive security reviews and is robust. It is assumed that the keys are securely provisioned on the device, the Join Server, and Application Server (and perhaps the Network Server). An attacker is likely to try to compromise the system at its weakest point, which is often when the keys are provisioned to the manufacturer, delivered to the operator, or stored in the operator's systems.

The keys SHALL be stored in a secure environment with a strictly limited access policy. The delivery of keys to an authorized party (for instance, when providing new devices to a customer) SHALL use secure methods and be sent to a very small distribution. It is particularly important the customer/operator store keys in a secure manner and limit their distribution. Many security breaches can be traced to allowing operations staff access to key material, so access to keys SHALL be severely restricted.

For organizations and/or IoT applications that are particularly sensitive, it should be mentioned there are a variety of solutions on the market that offer additional layers of security for LoRaWAN® systems. For example, if an end device is subject to physical threats, its keys can be protected in tamper resistant storage via a so-called Secure Element. In the network backend, an HSM (Hardware Security Module) is a dedicated piece of hardware specifically designed to manage the cryptographic key lifecycle.

2.6 ABP Devices

ABP devices are provisioned with the information required for network connectivity without going through the OTAA Join Procedure. As such, they are provisioned with a DevAddr and session keys. In addition, the LoRaWAN® specifications require the end-device to be provisioned with a DevEUI to disambiguate its identity to the network. It is recommended that the end-device be provisioned with all OTAA elements (DevEUI, JoinEUI and Root Keys). This allows the ABP device to operate as an OTAA device if required and provides a robust, internal mechanism for generating session keys from statically provisioned DevNonce and JoinNonce values.

Due to the vastly superior provisioning simplicity, dynamic address and session key generation and more robust security, OTAA is preferred to ABP in all but two scenarios:

1 When the end-device does not implement a radio receiver (such a device is not capable of being LoRaWAN® compliant).
2 When the end-device is so power constrained that the small number of JoinRequest/JoinAccept messages required to join the network constitutes a significant portion of its total lifetime power budget.

2.6.1 DevAddr

End-devices are assigned a 32-bit ephemeral device address (DevAddr) by the network operator when they join a network. ABP devices SHALL use a DevAddr assigned to them by the network operator to which they will connect or SHALL use a DevAddr allocated from one of the private NetID ranges.

Assigning a random or incorrect DevAddr would result in end-devices failing to be provisioned on the intended operator, failing to roam, or causing unintended traffic to be handled by other networks.

NetIDs are assigned by the LoRa Alliance® to member companies in good standing, which request them. Member companies commit to only forward traffic to the assigned owner of any given NetID.

2.6.2 Session Keys

ABP devices are provisioned with the session keys typically generated through the Join Process. A full set of these keys is required to be provisioned both in the end-device and in the network to allow end-device operation. ABP devices are provisioned with the session keys whereas OTAA devices generate them through the Join Process. Keys SHOULD be generated and handled using recommendations detailed in the OTAA root key section above.

2.6.3 Physical Provisioning

It is recommended that the end-device DevEUI be marked on the outside cover and easily legible. The LoRa Alliance® has published technical recommendations [TR0005] that defines a method of universal machine reading of the provisioning details which can be used to enable automated on-boarding. It is recommended that some type of machine-readable marking of the basic provisioning information (DevEUI and JoinEIU) exist on the device exterior to facilitate rapid and reliable device deployment. Security keys SHALL NOT be printed or displayed in any way on the end-device.

3 LoRaWAN Protocol Stack

3.1 Join Procedure

3.1.1 Description

Only an end-device SHALL initiate the Join Procedure. Use of the JoinRequest SHOULD only be used after commissioning; a factory reset; loss of connectivity with the network; when one of the frame counters has reached its maximum value; or new session keys are generated for security reasons.

Join requests create additional work for the Network Server, Join Server and Application Server as well as additional network traffic to exchange key material, etc. Limiting Join requests will reduce unnecessary downlink and MAC command traffic.

3.1.2 Recommended Practice

If the device is powered down completely, it SHOULD store the derived NwkSKey, AppSKey, frame counters and other RF parameters for use when it powers up again. For a static device there may be a Network Operator requirement to occasionally trigger a Join Request message allowing them to regenerate their network and application session keys, but this SHOULD be limited to once a month at most.

Transmission of the JoinRequest is governed by the retransmission backoff procedure specified in [TS001]. This includes the case when the end-device initiates the Join Procedure due to the temporary loss of power. In some end-devices as the power system near depletion, the end-device retains enough charge to transmit an uplink (JoinRequest) but the act causes a brown-out reset of the host CPU/MCU. The end-device SHALL guarantee that the backoff duty-cycle is observed even in this condition.

In LoRaWAN® 1.1+, the network MAY request the end-device initiate the Join Procedure, which may be required to refresh session keys or to reallocate DevAddr. As LoRaWAN® 1.0.x does not support this ability, it is recommended that the end-device support an application-layer means of doing so.

3.2 Fixed Channel Plan Join Process Optimization

3.2.1 Description

The gateway in range of the end device may be operating on 64 or more channels or on fewer channels due to limitations of the deployed gateway radios. Optimizing the Join process will typically allow the end device to connect in fewer attempts than when using a purely random or purely sequential channel selection process.

3.2.2 Recommended Practice

In the fixed channel regions of US915 and AU915, there are eight eight-channel banks in a 64-channel gateway, each with eight 125 KHz channels plus one 500 KHz channel. Join Request attempts using US915 SHOULD be at DR0 for the 125 KHz channels and DR4 for the 500 KHz channels. For AU915 these Join Requests SHOULD be sent at DR2 and DR6 respectively. Join Request attempts SHOULD select one of the 8 channel banks and then randomly select a channel within that channel bank. A channel from each distinct channel bank SHOULD be used without repeating bank usage. After a single channel has been attempted from each channel bank, another cycle of attempts SHOULD be made using one of the remaining seven unused channels from each bank, again in random fashion. This continues until all channels in each bank are used once in a cycle of 72 Join Request attempts.

3.3 Detecting Loss of Network Connectivity

3.3.1 Description

An end-device may lose connectivity with a network for a variety of reasons:

1. The end-device has moved and is no longer in coverage either while using its current RF configuration or regardless of its RF configuration
2. The network conditions have changed (new interference, gateways have been reconfigured, etc.), resulting in loss of coverage
3. The network has lost state synchronicity with the end-device (session keys were lost, Frame Counters are out-of-sync, etc.)
4. The network operator has ceased operating the network

The end-device may use several techniques to detect and mitigate these conditions.

3.3.2 Recommended Practice

End-devices with Adaptive Data Rate (ADR) enabled (both the end-device and the network assert the ADR bit) SHALL follow the ADR back off procedure described in [TS001]. This procedure uses ADRAckReq and the receipt of Class A downlinks to test connectivity and will gradually restore the end-device's default ADR configurations in a stepwise manner. Ultimately the end-device is configured to use all default channels at maximum transmit power and lowest data rate. For fixed channel plan regions, this requires that ALL channels are enabled. For dynamic plan regions, this requires that the default (or join) channels are enabled in addition to any dynamically configured channels. ADRAckReq remains set, however, if after an extended period after returning to default ADR configuration (an additional ADR_ACK_LIMIT uplinks sent with no Class A downlink received is strongly RECOMMENDED) and no Class A downlinks have been received, the end-device MAY consider network connectivity to be lost.

End-devices not being controlled through ADR MAY test network connectivity in a similar manner but SHALL use a technique other than ADRAckReq to request a downlink from the network. If these methods are not part of the end-device's normal operating mode, it is RECOMMENDED that they be used only occasionally (in the same manner ADRAckReq is used). End-devices that employ these methods as part of their normal operation but are also controlled through ADR MAY use these techniques to augment the ADRAckReq based backoff. Several examples of such techniques are:

1. Send a confirmed uplink (which may contain no application payload), which requests an acknowledgement from the network.
2. Send a LinkCheckReq, which requests an LinkCheckAns from the network.
3. Send an application-layer message which requests and application-layer response in a Class A window.

It is further recommended that end-devices only deploy these techniques in a manner that mirrors the ADR back off algorithm:

1. The end-device reverts to default ADR and Channel configuration in a stepwise manner.
2. Any Class A downlink halts the back off procedure and confirms network connectivity.
3. The network is given multiple attempts to send the confirming Class A downlink during each back off step.
4. Loss of connectivity is only determined after multiple attempts at default RF configuration – keep in mind that in many fixed channel plan regions, only 1/8th of transmitted messages may be received by some gateways – the determination of loss of connectivity MUST take this into account and not prematurely assume connectivity is lost.

3.4 Reacting to Loss of Network Connectivity

3.4.1 Description

If an end-device determines it is no longer in contact with the network (using the procedures described above) it MAY take additional steps to mitigate connectivity issues not related to RF coverage (which are addressed above). These root causes are:

1. Loss of state synchronicity between the end-device and the network (session keys or Frame Counters)
2. Cease in operation or contractual agreement between the end-device owner and the network operator and the need to establish connectivity with a new operator

ABP end-devices have no recourse in these situations, thus the recommendations are scoped to OTAA end-devices.

3.4.2 Recommended Practice

End-devices which support LoRaWAN 1.1+ SHALL use ReJoin Type 1 messages to probe the network for an opportunity to synchronize end-device security and radio state (DevAddr, session keys, frame counters, channel plans and downlink RF parameters). This technique mitigates both cases described above without impacting the end-device’s ability to continue to deliver uplink messages.

End-devices which support versions of LoRaWAN prior to 1.1 MAY use JoinRequest messages to attempt to restore connectivity in these cases. While this technique mitigates both cases described above, it does so at the expense of the end-device’s ability to continue to deliver uplink messages. Once the end-device issues a JoinRequest, uplink messages are no longer valid until a valid JoinAccept is received and processed. In the case of downlink-only connectivity issues, the act of sending the JoinRequest will cause the end-device to no longer be able to transmit any uplink messages until a valid JoinAccept is received. As a result, the decision to send a JoinRequest while a security session is already established SHOULD only be used as a last resort. As fully specified in [TS001], end-devices SHALL comply with the retransmission backoff behavior during an attempt to re-establish connectivity with the network through the Join Procedure.

3.5 Rolling Session Keys

3.5.1 Description

The end-device, the network or the application may determine that the session keys for the end-device's current security session need to be refreshed. This may be due to suspected security compromises, the lifetime of the existing session, that Frame Counters are approaching maximum values or other security considerations.

The ability to re-key the end-device is limited to OTAA end-devices.

3.5.2 Recommended Practice

For end-devices which support LoRaWAN 1.1+, the end-device SHALL initiate a re-keying event through the transmission of a ReJoin Type 2 message. Network-side re-keying SHALL be initiated via the ForceReJoinReq MAC command. Application re-keying MAY be initiated via application-layer messaging or through an interface to the Network Server requesting it to send the ForceReJoinReq MAC command. This enables the graceful re-keying of the-end-device without interrupting the delivery of uplink messages.

End-devices which support versions of LoRaWAN prior to 1.1 MAY use JoinRequest messages to roll session keys. This technique re-keys the device at the expense of the end-device's ability to continue to transmit uplink messages. Once the end-device issues a JoinRequest, uplink messages SHALL NOT be transmitted until a valid JoinAccept is received and processed. In the case of downlink-only connectivity issues, the act of sending the JoinRequest will cause the end-device to no longer be able to deliver any uplink messages. As a result, the decision to send a JoinRequest while a security session is already established SHOULD only be used as a last resort. There is no network-side facility to initiate a re-keying event. It is recommended that application layer re-keying initiation be supported by LoRaWAN® 1.0.x end-devices.

3.6 DevNonce Values SHALL Not Be Reused

3.6.1 Description

LoRaWAN® requires DevNonce values (for any given DevEUI/JoinEUI pair) never be reused. The simplest and most effective way to accomplish this is for the end-device to increment the DevNonce for each JoinRequest transmitted, starting at zero for the first JoinRequest.

Join Servers keep a limited history of DevNonce values previously used by an end-device to detect a JoinRequest replay and will reject a JoinRequest if it contains DevNonce which has been previously used. Using an incrementing DevNonce enables the Join Server to practically detect all JoinRequest replays instead of only being able to detect replays of a limited number of random DevNonces.

LoRaWAN v1.0.4 and v1.1.x make this behavior mandatory.

3.6.2 Recommended Practice

The end-device SHALL implement a strictly monotonically incrementing counter as the DevNonce. It is further RECOMMENDED that the end-device manufacturer notify the Join Server operator that the incrementing DevNonce mechanism is in place to enable the operator to modify the replay detection algorithm.

3.7 Avoiding Synchronous Behavior

3.7.1 Description

To maximize network capacity and message receipt success it is important that end-devices implement randomness in both the transmission timing and channel selection. Randomness in these two dimensions helps prevent collisions and uplink spikes which can artificially limit gateway and network capacity.

3.7.2 Recommended Practice

End-devices SHALL create a pseudo-randomly sorted list of enabled channels and iterate through this list with each transmission. The list SHALL be re-sorted when new channels are enabled or disabled. The end-devices SHALL ensure that the pseudo-random number generators used to create this list are seeded with truly random values to prevent multiple end-devices from systematically using identically sorted lists. Some regulatory regions require every channel to be used equally and this technique also meets that requirement.

End-devices SHALL add a pseudo-random delay to all periodic transmissions (including JoinRequests and retransmissions controlled by `NbTrans` or the application-layer). The end-device SHALL ensure that the pseudo-random number generators used to create this list are seeded with truly properly random values (see RFC4086 for examples and methodology) to prevent multiple end-devices from systematically using the same delay values.

Groups of end-devices SHOULD never be coordinated to transmit at the same real-time. Even loose coordination of transmit time places undue burden on the network. An example of behaviors to be avoided are:

1 Daily uplinks at midnight of a day's worth of sensor readings
2 Weekly validation of the configuration of a fleet of devices to occur with a 12-hour window every week
3 Transmitting subsequent confirmed uplinks, having not received an acknowledgement at exactly 2500 mSec after the previous uplink

3.8 Avoiding Congestion Collapse

3.8.1 Description

A network congestion collapse occurs when an event triggers widespread and repeated communication attempts. For example, many end-devices use a reed-switch for basic, non-contact control (power on/off, re-Join, etc.); should an earthquake occur near a large population of end-devices, it is probable that they will all simultaneously reboot and initiate the Join Procedure. The fact that these messages are all sent simultaneously could overwhelm the gateway front ends, the gateway backhauls, or even the Network Server or Join Server. In which case, no response would be sent, causing the end-devices to again send the JoinRequest. In the worst case, the end-devices react by sending the requests at an even faster pace to try to quickly receive a response. When large groups of end-devices behave this way, it can lead to a condition in which the network cannot service any of them.

There are many such trigger events which can impact any end-device and any network, some first order (like an earthquake or power outage) others second or third order (a chain of conditions that ultimately lead to the collapse). As a result, mitigation techniques need to be universally implemented.

Any time the end-device sends an uplink which requires a response (ACK, JoinAccept, MAC Command response, Application-Layer response, etc.) an opportunity for congestion collapse is introduced. If the end-device reacts to the lack of response by sending another uplink which again requires a response, then it SHALL implement techniques to avoid instigating a congestion collapse.

Even end-device communications which do not demand a response can lead to temporary collapse if the end-device enters a mode where it dramatically increases the pace at which it sends uplinks for an extended period of time.

3.8.2 Recommended Practice

Whenever possible, end-devices SHOULD use communication techniques which do not require responses. In addition to avoiding congestion collapse, this technique also preserves the downlink capacity of the network, which for LoRaWAN networks is less than the uplink capacity.

End-devices SHALL use the randomization techniques of both transmit time and channel described above to avoid synchronous transmissions among groups of end-devices. This will help mitigate the temporary congestion possible even when responses are not requested.

> In response to a missed response, the end-device SHALL decrease the pace of transmission (increase the periodicity of transmission). A reasonable back off strategy described in [TS001-1.0.4] is reproduced here:
>
> *Uplink frames that*
>
> - *require an **acknowledgment or answer** from the Network or an Application Server and are **retransmitted** by the end-device if the acknowledgment or answer is not received.*
>
> ***and***
>
> - *can be triggered by an **external** event causing **synchronization** across a large (>100) number of end-devices (power outage, radio jamming, network outage, earthquake…)*
>
> *can trigger a catastrophic, self-persisting, radio network overload situation.*

> ***Note:*** *A typical example of such an uplink frame is a Join-Request if the implementation of a group of end-devices decides to reset the MAC layer in the case of a network outage. The entire group of end-devices will start broadcasting Join-Request uplinks and will stop only upon receiving a Join-Accept from the Network.*
>
> *For those frame retransmissions, the interval between the end of the RX2 slot and the next uplink retransmission SHALL be random and follow a different sequence for every end-device (for example using a pseudo-random generator seeded with the end-device's address). The transmission duty-cycle of such a frame SHALL respect local regulations and the following limits, whichever is more constraining:*

Aggregated during the first hour following power-up or reset	$T_0 < t < T_{0+1}$	*Transmit time* *< 36 s per hour*	*1% duty cycle*
Aggregated during the next 10 hours	$T_{0+1} < t < T_{0+11}$	*Transmit time* *< 36 s per 10 h*	*0.1% duty cycle*
After the first 11 hours, aggregated over 24 h, where N refers to days starting at 0	$T_{0+11} + N \times (24\ \text{hours/day}) < t < T_0 + 35 + N \times (24\ \text{hours/day}),\ N \geq 0$	*Transmit time* *< 8.7 s per 24 h*	*0.01% duty cycle*

Table** 1: **Transmit duty-cycle limitations

Additional back off algorithms MAY be implemented within each period described above.

3.9 Discontinuing Retransmissions

3.9.1 Description

End-devices will sometimes issue retransmissions (uplink messages with the same content and Frame Counter as the previous message) to improve message delivery. This is controlled via the `NbTrans` parameter. Any reception by the end-device in a Class A window explicitly indicates that the message has been received by the network and no additional retransmissions of the same Frame Counter are required.

3.9.2 Recommended Practice

When an end-device retransmits an uplink message multiple times because of `NbTrans` > 1, it SHALL stop retransmitting the message as soon as it has received any valid Class A downlink. This behavior is fully specified and required by both LoRaWAN 1.0.4+ and LoRaWAN 1.1+.

3.10 Default Channels

3.10.1 Description

There are two styles of regional channel plans defined in [RP002]: dynamic and fixed. Default channels are the channels that are required to be implemented on the end-device for each region. These channels may be disabled by the network, but all end-devices begin operation using the default channels and re-enable all default channels as the last step of ADR back off.

Dynamic channel plan regions (EU868, CN779, EU433, IN865, KR920, AS923-1, AS923-2, AS923-3 as of [RP002-1.0.2]) define a small number of default channels (two or three depending on region). Additional channels are dynamically created by the network via the `CFList` or ***NewChannelReq*** commands.

For fixed channel plan regions (US915, AU915, CN470 as of [RP002-1.0.2]) a large set of default, fixed channels are defined. Channels are enabled and disabled via the `CFList` or ***LinkADRReq*** commands. In a future version of the LoRaWAN Layer 2 specification, fixed channel plan regions will also support the ***NewChannelReq*** command for defining new channels.

Networks may be deployed with gateways which support a variety of different channels, both in terms of number of channels and frequencies supported. To be able to use these various network configurations end-devices need to correctly implement default channel behaviors.

3.10.2 Recommended Practice

End-devices SHALL begin operation with all default channels enabled. OTAA devices will use all these channels for the Join Procedure. As the last step of ADR back off, the end-device SHALL enable all default channels.

End-devices SHALL support the full range of channel indexes defined in [RP002]

These behaviors are clearly specified in [TS001-1.0.4] and later specifications.

3.11 FcntUp/FcntDown SHALL Use 32 Bits

3.11.1 Description

The LoRaWAN® Layer 2 Specifications prior to version TS001-1.0.4 allow for the use of a 16-bit or 32-bit wide Frame Counters. To maximize end-device security and reduce the risk of a message playback attacks, all implementations SHALL use 32-bit wide counters.

3.11.2 Recommended practice

End-devices SHALL implement 32-bit unsigned integer counters for all Frame Counters.

[TS001-1.0.4] and later specifications mandate 32-bit wide frame counters.

3.12 Frame Counters Incremented only with Frame Transmission

3.12.1 Description

The Frame Counter is used for both security (for MIC generation/validation and encryption/decryption) and detection of lost frames. By only incrementing the Frame Counter after transmission, end-devices, Application Servers, and Network Servers may reliably use the Frame Counter to detect lost frames. In turn this information may influence RF configuration to optimize communication reliability and efficiency.

3.12.2 Recommended Practice

End-devices, Application Servers, and Network Servers SHALL only increment (by one) Frame Counters after a message transmission has been attempted.

3.13 OTAA Required Persistent Values

3.13.1 Description

End-devices operating in OTAA mode are required to persist several values to remain compliant with [TS001].

3.13.2 Recommended Practice

For end-devices which will initiate a Join Procedure following a reset or loss of power, the following values SHALL be persisted. End-devices which will NOT initiate a Join Procedure following a reset or loss of power SHALL persist these values as well as those described for ABP end-devices.

- **DevEUI**
- **JoinEUI**
- **Root Keys (AppKey/NwkKey)**
- **DevNonce**
  The DevNonce SHALL be implemented as a counter, the last value used SHALL be persisted to ensure no previous values are re-used.
- **JoinNonce**
  The JoinNonce needs to be validated against reuse by the end-device. If the JS supports incrementing JoinNonce values, as recommended in [TR001], a last value SHALL be persisted for validation. If the JS does not support incrementing JoinNonce values the end-device SHOULD persist a reasonable number of most recently seen values.

From version [TS001-1.0.4], DevNonce SHALL be implemented as a counter.

3.14 ABP Required Persistent Values

3.14.1 Description

End-devices operating in ABP mode are required to persist several values to remain complaint with [TS001] and ensure that they remain in contact with the network.

3.14.2 Recommended Practice

All ABP end-devices SHALL persist the following values through a reset or loss of power.

- **DevEUI**
- **DevAddr**
- **SessionKeys**
- **Frame Counters** (both uplink and downlink)

End-devices which do not support ResetInd SHALL persist the additional values through a reset or loss of power.

- **Channel List** (frequencies and enabled channels)
- **Data rate**
- **TxPower**
- **NbTrans**
- **MaxDutyCycle**
- **RX2Frequency**
- **RX1DROffset**
- **RX2DataRate**
- **RXTimingDelay**
- **MaxEIRP**
- **DownlinkDwellTime**
- **UplinkDwellTime**

If an end-device is operating with Class B enabled, the following additional values SHALL be persisted through a reset or loss of power.

- **PingSlotPeriodicity**
- **BeaconFrequency**
- **PingSlotFrequency**
- **PingSlotDataRate**

3.15 ADR (Adaptive Data Rate) Support

3.15.1 Description

ADR is used by the network server to optimize the configuration of the end-device to:

1 Maximize connectivity
2 Minimize airtime
3 Minimize end-device power consumption

ADR controls the channel plan, data rate, transmit power and number of retransmissions used by the end-device. These attributes are dynamically adapted as the network observes changes in the RF conditions and gateway configurations supporting the end-device.

In all cases, the network is in the best position to determine the optimum ADR configurations for the end-device as it enjoys a network-wide view, where the device has no visibility outside of its own behavior. This is true even for mobile devices.

3.15.2 Recommended Practice

ADR (Adaptive Data Rate) SHOULD be fully supported and enabled by default. The end-device owner SHOULD coordinate with the network operator to define a device management schema (i.e. profile) which optimizes its connectivity. This is particularly true for end-devices with special considerations (mobile, nomadic, or other widely varying RF conditions).

3.16 ADR Back Off

3.16.1 Description

[TS001] described the mechanism by which ADR controlled end-devices restore connectivity by executing a stepwise algorithm that restores maximum transmit power, gradually decreases data rate and ultimately re-enables all default channels. The pace of this back off is controlled by a pair of parameters: ADR_ACK_LIMIT and ADR_ACK_DELAY. By default, ADR_ACK_LIMIT is 64 uplink messages without a Class A downlink and ADR_ACK_DELAY is 32 uplink messages without a Class A downlink.

3.16.2 Recommended Practice

End-devices SHOULD implement ADR and the ADR back off algorithm with the default values of ADR_ACK_LIMIT and ADR_ACK_DELAY, defined in [RP002]. If the end-device implements values other than the default for ADR_ACK_LIMIT and ADR_ACK_DELAY, the end-device SHALL communicate these values to the network operator out of band, during the provisioning process.

3.17 Duty Cycle Limitations

3.17.1 Description

[TS001] defines some duty cycle limitations and controls over the end-device to improve network optimization, however, regulatory regions around the world may enforce specific duty cycle limitations in addition to those specified by LoRaWAN®. These limitations may be different by frequency band, channel bandwidth or transmit power. Duty cycle limitations may apply to the gateway as well as the end-device, which may impact the network's ability to send downlinks in response to requests from the end-device.

3.17.2 Recommended Practice

The end-device SHALL respect the lower of the two duty cycle limitations (LoRaWAN or Regulatory). Most often, a LoRaWAN network will not set a duty cycle limitation, and the end-device SHALL respect the regulatory duty cycle limitations

3.18 Transmit Power

3.18.1 Description

To efficiently support the greatest variety of operating conditions it is useful for the end-device to implement transmit power control across the widest reasonable range of power levels which are specified in [RP002].

Different regulatory regions support a variety of maximum transmit power levels. These may be specified as antenna port conducted power, EIRP, power-spectral-density or other measured values.

3.18.2 Recommended Practice

End-devices SHOULD support the highest practical transmit power in the region they are certified to operate in.

End-devices SHOULD support the full range of power control specified in [RP002].

End-device manufacturers designing devices targeted (or which may be targeted in future) for multiple regulatory regions SHOULD design them to support the maximum practical transmit power across those regions.

From version [TS001-1.0.4], a minimum power control range is mandatory. An end-device SHALL support a minimum power control range of 14dB, or support a minimum transmit power of 2dBm.

3.19 Class B/C Cell Update

3.19.1 Description

End-devices operating in Class B or Class C mode may need to occasionally inform the network of which gateway (cell) to use for downlinks. While Class A uplinks continuously inform the network as to which gateways are best used for downlinks, end-devices receive Class B/C downlinks without an immediately preceding uplink. As the RF conditions or network change or if the end-device moves, it is necessary for the end-device to send an uplink to inform the network that a new set of gateways may need to be selected for downlinks, however, it is best to minimize the number of uplinks used solely for this purpose.

If the network is not informed of the end-device's new cell location, Class B and Class C downlinks may be transmitted by gateways which are not within range of the end-device.

3.19.2 Recommended practice

End-devices MAY use Class B beacons (regardless of the end-device Class) to detect changes in the set of gateways connecting it to the network. Class B beacons MAY contain gateway specific fields, including GPS coordinates of the gateway, a NetID and GatewayID or other proprietary extensions. If this technique is used to detect cell changes, the end-device SHOULD keep a list of recent gateway identifiers. This list SHOULD be sized to track a reasonable number of local gateways. If a new gateway identifier is added to the list the end-device SHOULD send an uplink frame to allow the network to update its cell location. The end-device SHOULD age entries out of this list if no beacon containing the entry's gateway id has been received in a long period of time.

For end-devices operating in areas where beacons (or beacons with gateway specific fields identifying the gateway) are not available, cell change detection is not possible. If the end-device includes an accelerometer, GPS/GNSS or other means to detect that it has moved to a new location, the end-device MAY elect to transmit an uplink frame to inform the network of a possible cell change. The end-device SHOULD send such a frame only if it detects that a sufficiently large change in location may have occurred.

All end-devices operating in Class B or Class C mode SHOULD send periodic uplink messages to keep the network informed of its cell location. The periodicity of these messages MAY be very low if the above event triggered messages are also implemented.

4 Glossary

AS	Application Server
FUOTA	Firmware Update Over-The-Air
JS	Join Server
JoinEUI	The address of the Join Server
AppEUI	Deprecated, the address of the Join Server, now called JoinEUI
NS	Network Server
OUI	Organization Unique Identifier

5 NOTICE OF USE AND DISCLOSURE