LoRaWAN™ Backend Interfaces 1.0 Specification

NOTICE OF USE AND DISCLOSURE

LoRa Alliance, Inc.
3855 SW 153rd Drive
Beaverton, OR 97003

LoRaWAN™ Backend Interfaces 1.0 Specification

Authored by the LoRa Alliance Technical Committee

Chairs:
N.SORNIN (Semtech), A.YEGIN (Actility)

Editor:
A.YEGIN (Actility)

Contributors:
A.BETOLAUD (Gemalto), E.BRUINZEEL (KPN), P.CHRISTIN (Orange), P.COLA (Bouygues Telecom), P.DUFFY (Cisco), F.DYDUCH (Bouygues Telecom), J.ERNST (Swisscom), E.FORMET (Orange), O.HERSENT (Actility), D.KJENDAL (Senet), M.KUYPER (TrackNet), M.LEGOURRIEREC (Sagemcom), C.LEVASSEUR (Bouygues Telecom), M.PAULIAC (Gemalto), N.SORNIN (Semtech), P.K.THOMSEN (Orbiwise), A.YEGIN (Actility)

Version: 1.0
Date: October 11, 2017
Status: Final release

Contents

Tables

Figures

1 Introduction

This document describes the standard interfaces and message flow between

1. A Network Server and a Join Server
2. A Join Server and an Application Server
3. Two Network servers in the case of roaming traffic routing

The Network Server to Application Server interface is outside the scope of this document.

The primary focus of this document is to describe the message flow between the various entities of the network during the Over-the-Air Activation and Roaming Procedures of an End-Device.

2 Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

3 Network Reference Model

Figure 1 and Figure 2 show the Network Reference Model (NRM) for the LoRaWAN architecture.

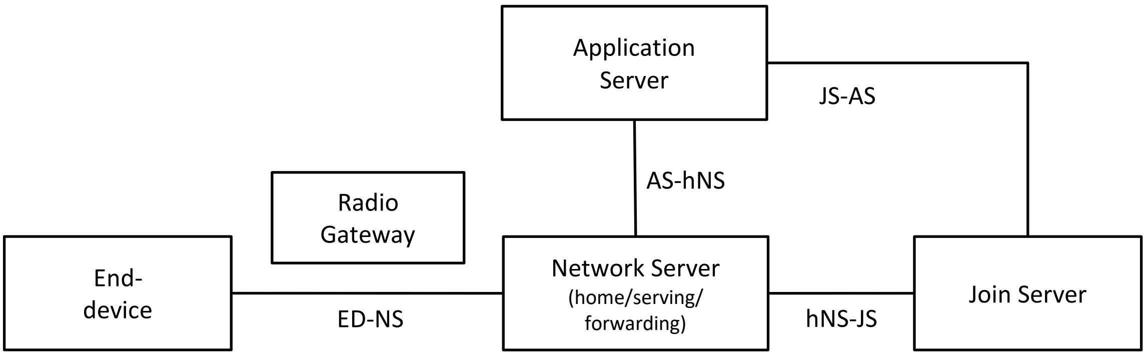

Figure 1 LoRaWAN Network Reference Model (NRM), End-Device at home

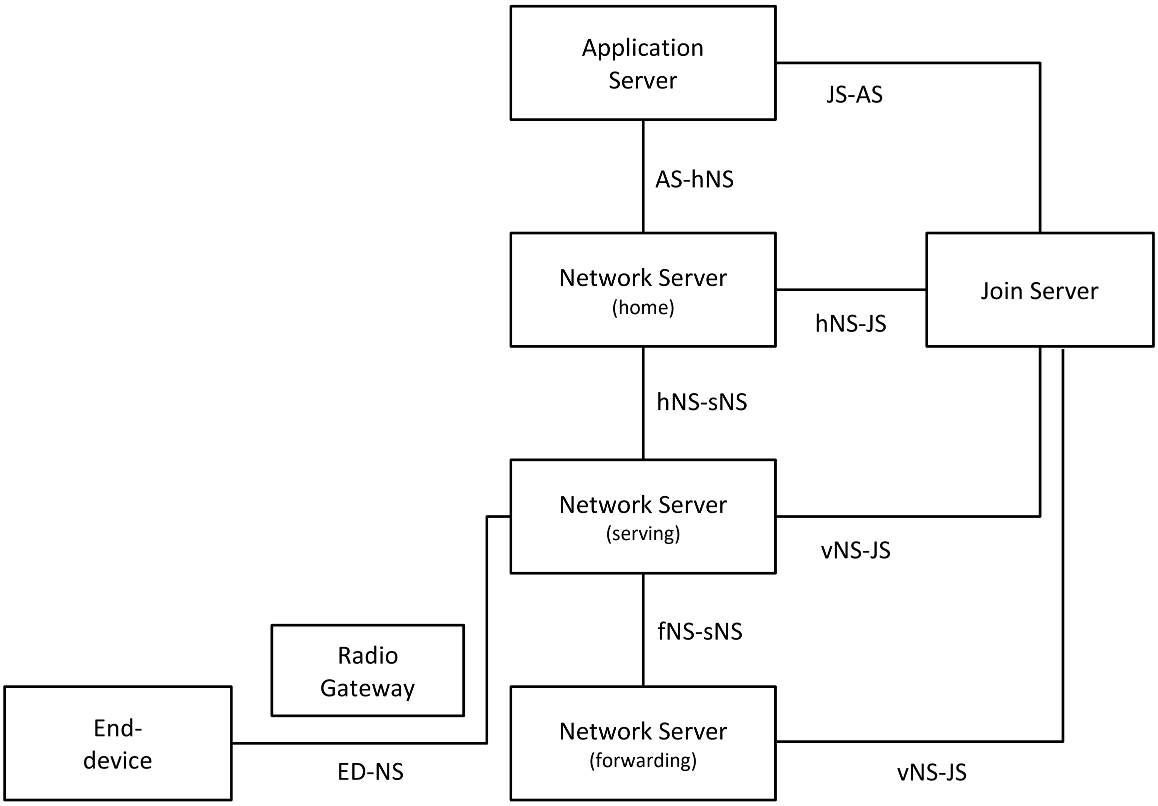

Figure 2 LoRaWAN Network Reference Model (NRM), roaming End-Device

End-Device:

The End-Device is a sensor or an actuator. The End-Device is wirelessly connected to a LoRaWAN network through Radio Gateways. The application layer of the End-Device is connected to a specific Application Server in the cloud. All application layer payloads of this End-Device are routed to its corresponding Application Server.

Radio Gateway:

The Radio Gateway forwards all received LoRaWAN radio packets to the Network Server that is connected through an IP back-bone. The Radio Gateway operates entirely at the physical layer. Its role is simply to decode uplink radio packets from the air and forward them unprocessed to the Network Server. Conversely, for downlinks, the Radio Gateway simply executes transmission requests coming from the Network Server without any interpretation of the payload.

Network Server:

The Network Server (NS) terminates the LoRaWAN MAC layer for the End-Devices connected to the network. It is the center of the star topology.

Generic features of NS are:

- End-Device address check,
- Frame authentication and frame counter checks,
- Acknowledgements,
- Data rate adaptation,
- Responding to all MAC layer requests coming from the End-Device,
- Forwarding uplink application payloads to the appropriate Application Servers,
- Queuing of downlink payloads coming from any Application Server to any End-Device connected to the network,
- Forwarding Join-request and Join-accept messages between the End-Devices and the Join Servers.

In a roaming architecture, an NS may play three different roles depending on whether the End-Device is in roaming situation or not, and the type of roaming that is involved.

Serving NS (sNS) controls the MAC layer of the End-Device.

Home NS (hNS) is where Device Profile, Service Profile, Routing Profile and DevEUI of the End-Device are stored. hNS has a direct relation with the Join Server that will be used for the Join Procedure. It is connected to the Application Server (AS). When hNS and sNS are separated, they are in a roaming agreement. Uplink and downlink packets are forwarded between the sNS and the hNS.

Forwarding NS (fNS) is the NS managing the Radio Gateways. When sNS and fNS are separated, they are in a roaming agreement. There may be one or more fNS serving the End-Device. Uplink and downlink packets are forwarded between the fNS and the sNS.

Join Server:

The Join Server (JS) manages the Over-the-Air (OTA) End-Device activation process. There may be several JSs connected to a NS, and a JS may connect to several NSs.

The End-Device signals which JS should be interrogated through the JoinEUI field of the Join-request message. Each JS is identified by a unique JoinEUI value. Note that AppEUI field of the Join-request in LoRaWAN 1.0/1.0.2 [LW10, LW102] is renamed to JoinEUI field in LoRaWAN 1.1 [LW11]. The term JoinEUI is used to refer to the AppEUI in the context of LoRaWAN 1.0/1.0.2 End-Devices in this specification.

The JS knows the End-Device's Home Network Server identifier and provides that information to the other Network Servers when required by the roaming procedures.

The JS contains the required information to process uplink Join-request frames and generate the downlink Join-accept frames. It also performs the network and application session key derivations. It communicates the Network Session Key of the End-Device to the NS, and the Application Session Key to the corresponding Application Server.

For that purpose the JS SHALL contain the following information for each End-Device under its control :

- DevEUI
- AppKey
- NwkKey (only applicable to LoRaWAN 1.1 End-Device)
- Home Network Server identifier
- Application Server identifier
- A way to select the preferred network in case several networks can serve the End-Device
- LoRaWAN version of the End-device (LoRaWAN 1.0, 1.0.2, or 1.1)

The root keys NwkKey and AppKey are only available in the JS and the End-Device, and they are never sent to the NS nor the AS.

Secure provisioning, storage, and usage of root keys NwkKey and AppKey on the End-Device and the backend are intrinsic to the overall security of the solution. These are left to implementation and out of scope of this document. However, elements of this solution may include SE (Secure Elements) and HSM (Hardware Security Modules).

The way those information are actually programmed into the JS is outside the scope of this document and may vary from one JS to another. This may be through a web portal for example or via a set of APIs.

The JS and the NS SHALL be able to establish secure communication which provides end-point authentication, integrity and replay protection, and confidentiality. The JS SHALL also be able to securely deliver Application Session Key to the Application Server.

The JS may be connected to several Application Servers (AS), and an AS maybe connected to several JSs.

The JS and the AS SHALL be able to establish secure communication which provides end-point authentication, integrity, replay protection, and confidentiality.

Application Server:

The Application Server (AS) handles all the application layer payloads of the associated End-Devices and provides the application-level service to the end-user. It also generates all the application layer downlink payloads towards the connected End-Devices.

There may be multiple ASs connected to a NS, and an AS may be connected to several NSs (operating End-Devices through several networks, for example). An AS may also be connected to multiple JSs.

The Home NS routes the uplinks toward the appropriate AS based on the DevEUI.

In addition to the aforementioned network elements, LoRaWAN architecture defines the following network interfaces among these entities:

hNS-JS: This interface is used for supporting the Join (Activation) Procedure between the JS and the NS.

vNS-JS: This interface is used for Roaming Activation Procedure. It is used to retrieve the NetID of the hNS associated with the End-Device.

ED-NS: This interface is used for supporting LoRaWAN MAC-layer signaling and payload delivery between the End-Device and the NS.

AS-hNS: This interface is used for supporting delivery of application payload and also the associated meta-data between the AS and the NS.

hNS-sNS: This interface is used for supporting roaming signaling and payload delivery between the hNS and the sNS.

sNS-fNS: This interface is used for supporting roaming signaling and payload delivery between the sNS and the fNS.

AS-JS: This interface is used for delivering Application Session Key from the JS to the AS.

4 End-Device Types and States

There are two types of LoRaWAN End-Devices: Activation-by-Personalization (ABP) activated End-Devices, and Over-the-Air (OTA) activated End-Devices. ABP End-Devices are directly tied to a specific network by skipping the Join Procedure. OTA End-Devices perform Join Procedure to get activated on a selected network.

Figure 3 shows the two types of End-Devices and various End-Device states associated with the OTA End-Devices.

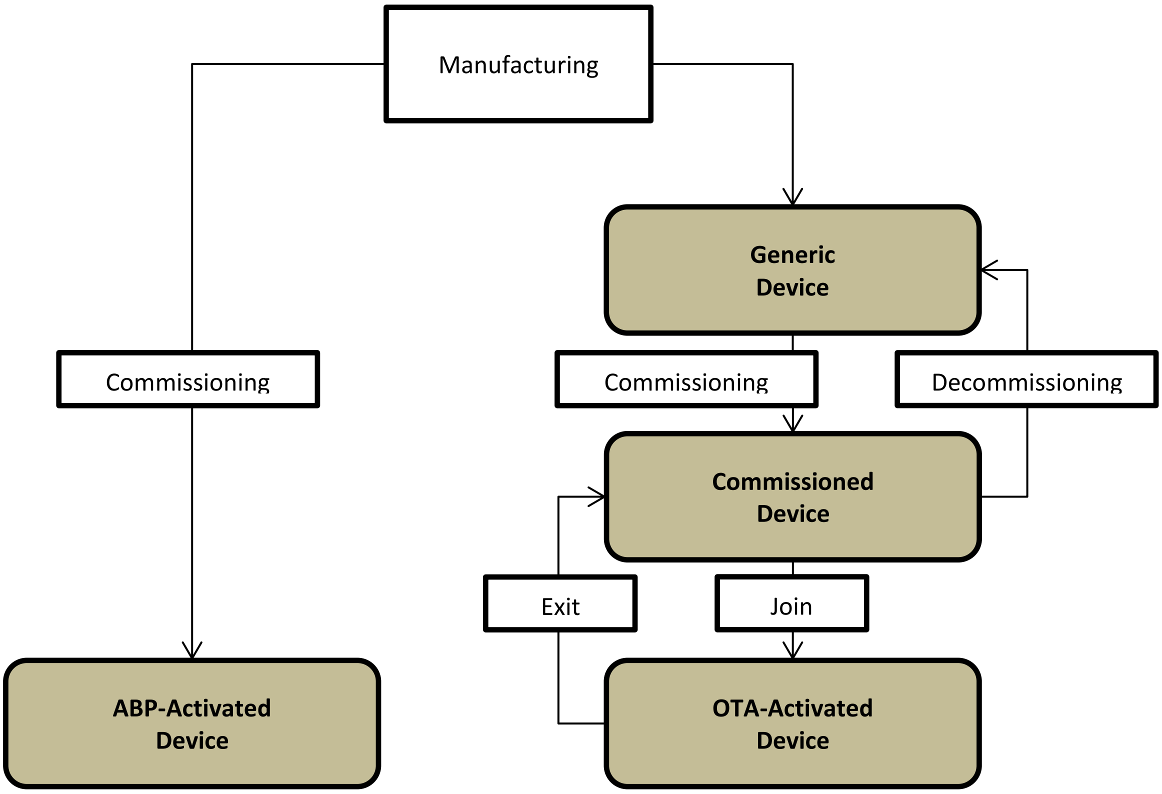

Figure 3 End-Device types and states

An ABP End-Device SHALL have the following information either when it leaves the manufacturer or upon configuration thereafter: DevAddr, AppSKey, network session keys. Network session keys are SNwkSIntKey, FNwkSIntKey, and NwkSEncKey in case of a R1.1, and NwkSKey in case of a R1.0/1.0.2 End-Device. For that End-Device to readily use the network, its Home NS SHALL have the DevAddr, network session keys, AS info of the End-Device; and the AS SHALL have the DevAddr, AppSKey of the End-Device.

An OTA End-Device SHALL have the following information either when it leaves the manufacturer or upon configuration thereafter: DevEUI, NwkKey (R1.1-only), AppKey, JoinEUI. At this point it is called a Generic End-Device. The associated JS SHALL have DevEUI, AppKey, NwkKey (R1.1-only) of the End-Device. No NS or AS may have any information about the Generic End-Device until it is commissioned.

Reconfiguration of an End-Device may be possible during its lifecycle. Configuration and reconfiguration procedure details are outside the scope of this specification.

Commissioning procedure associates the End-Device with its Home NS and a specific AS. The JS of a commissioned OTA End-Device SHALL have the Home NS info for the End-Device. The AS associated with the End-Device SHALL have the DevEUI of the End-Device. The Home NS SHALL have various profile information related to the End-Device and its service subscription. Mechanisms used for provisioning the AS, JS, and NS with the required information is outside the scope of this specification.

When a commissioned OTA End-Device performs successful Join (Activation) Procedure, it knows DevAddr, network session keys, and AppSKey. The JS knows the DevEUI, DevAddr, network session keys, AppSKey, and DevNonce. The JS delivers the DevEUI and AppSKey to the AS. The JS delivers the network session keys, and optionally the encrypted AppSKey to the NS.

5 Commissioning Procedure

Commissioning Procedure is executed by the AS, JS (only applicable to OTA), and NS for a given End-Device. It involves the JS associating the End-Device with a Home NS (only applicable to OTA), the Home NS and the AS receiving the profile information related to the End-Device and its service subscription. The mechanisms used for provisioning the required information on the aforementioned network elements is outside the scope of this specification.

Decommissioning Procedure breaks the association between the End-Device and the Home NS and the AS. This procedure involves resetting the state created on the AS and NS at the time of commissioning, unbinding the End-Device and Home NS on the JS (only applicable to OTA).

Details of the Commissioning and Decommissioning Procedures are outside the scope of this specification.

6 Activation of ABP End-Devices

Figure 4 shows activation of an ABP End-Device with an NS. This procedure applies to both R1.0 [LW10, LW102] and R1.1 [LW11] End-Devices and networks.

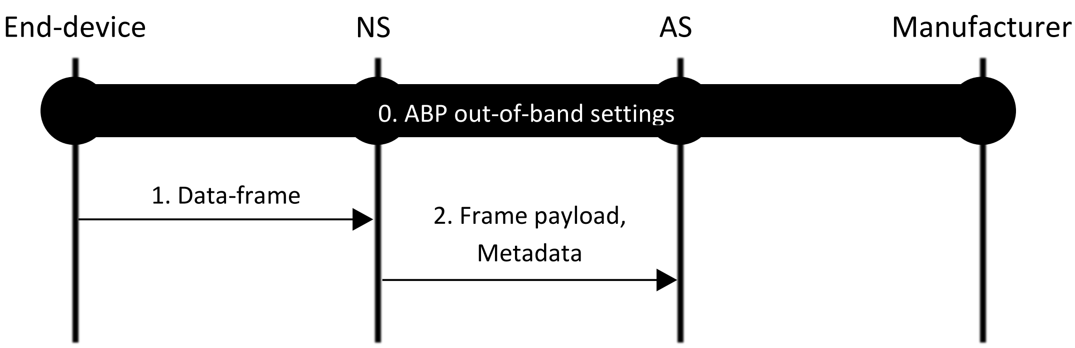

Figure 4 Activation of ABP End-Device

Step 0:

The End-Device, NS, and AS are configured with the required information, so that the End-Device can send packets as soon as it is powered on.

Step 1:

When the End-Device has application payload to send, it can do so without performing any setup signaling with the network. The packet includes application payload that is encrypted using the AppSKey, and the MIC that is generated using the network session integrity keys (SNwkSIntKey and FNwkSIntKey in case of a R1.1 End-Device, and NwkSKey otherwise).

When the NS receives the packet, it SHALL perform network session integrity key lookup based on the DevAddr of the received packet. The NS SHALL verify the MIC using the retrieved keys. If the keys are not found, or if the MIC verification fails, the NS SHALL drop the packet.

Step 2:

The NS SHALL send the encrypted payload of the accepted packet to the AS associated with the End-Device. The application payload may be accompanied with the metadata, such as DevAddr, FPort, timestamp, etc. The NS SHALL consider receipt of the very first packet from the End-Device as the activation of a LoRa session for the End-Device.

7 Activation of OTA End-Devices

OTA Activation Procedure is used by the End-Device in order to mutually authenticate with the network and get authorized to send uplink and receive downlink packets.

NSs are categorized in two ways with respect to an End-Device. Home NS is the NS that holds the End-Device, Service, and Routing Profiles of the End-Device, and interfaces with the AS and the JS after any activation. The mechanism used for provisioning the Home NS with the required profile information is outside the scope of this specification. On the other hand, Visited NS is any other NS that has a business and technical agreement with the Home NS for being able to serve the End-Device.

There are two variants of the Activation Procedure, namely Activation at Home, and Roaming Activation.

Activation at Home: The End-Device performs the Activation Procedure within the radio coverage of the Home NS. At the end of the procedure, the Home NS is the only NS serving the End-Device for reaching out to the AS and the JS.

Roaming Activation: The End-Device performs the Activation Procedure outside the radio coverage of its Home NS but within the coverage of a Visited NS. In this procedure, the Visited NS learns the identity of the Home NS with the help of the JS and obtains the required End-Device and Service Profiles from the Home NS. At the end of the procedure, the End-Device is served by both the Visited NS and the Home NS for reaching out to the AS and the JS.

When the End-Device performs a successful Join or Rejoin Procedure, the End-Device is said to have a LoRa session with the backend. Each LoRa session is associated with a set of context parameters managed on the End-Device, and the NS, JS, and AS. (e.g., session keys, DevAddr, ID of NS, etc.). The LoRa session terminates when the End-Device performs Deactivation (Exit) Procedure or another successful Join/Rejoin Procedure.

8 OTA Activation at Home Procedure

Figure 5. illustrates the message flow for OTA Activation at Home Procedure. This procedure applies to both R1.0 [LW10, LW102] and R1.1 [LW11] End-Devices and networks.

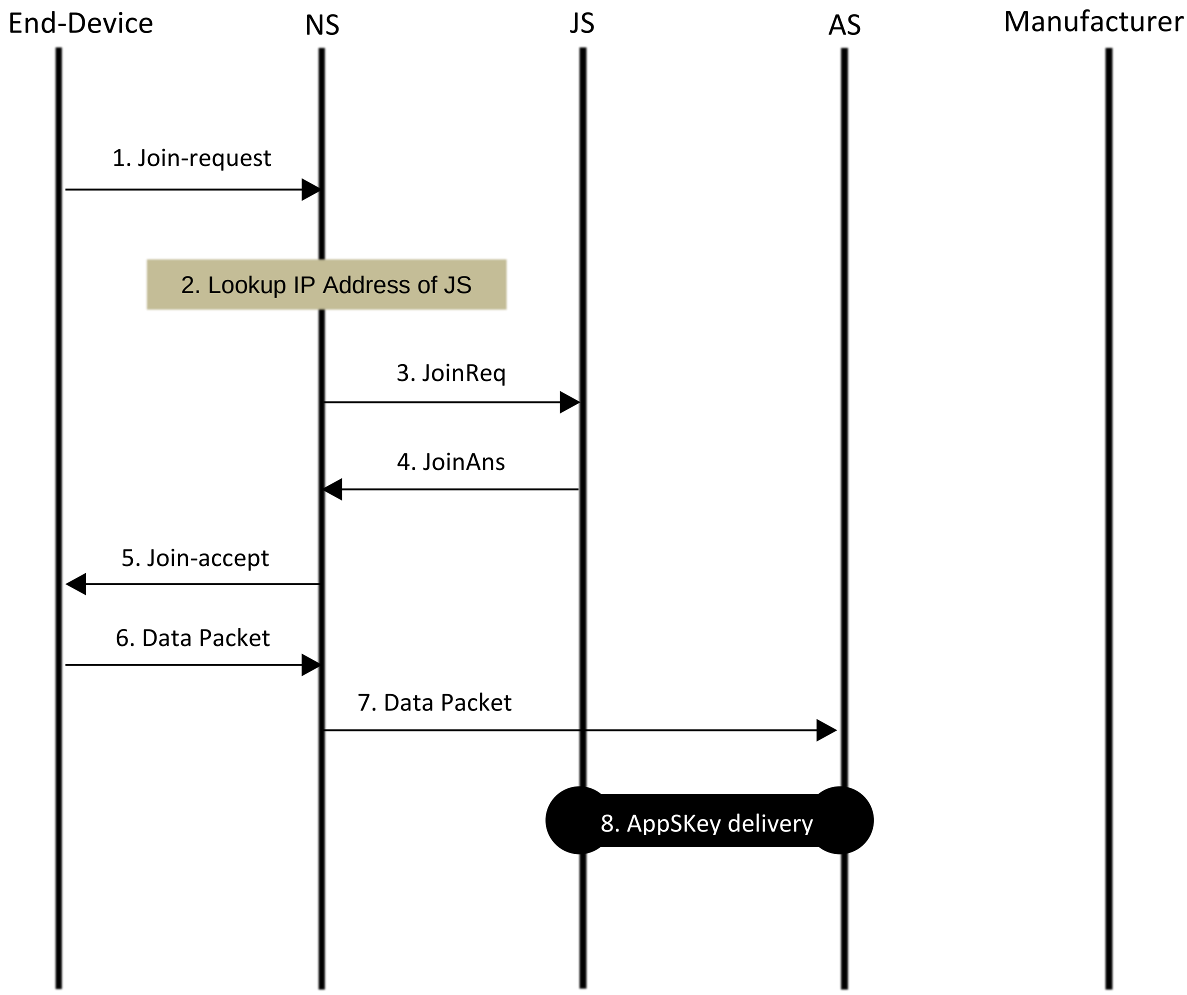

Figure 5 Message flow for OTA Activation at Home Procedure.

Step 1:

The End-Device SHALL transmit a Join-request message.

Step 2:

When the NS receives the Join-request message, the NS SHALL determine whether it is the Home NS for the End-Device identified by DevEUI, or not. In this flow it is assumed that the NS is the Home NS of the End-Device. See Section 12 for the case where the NS is not the Home NS of the End-Device, but the NS is configured to use the JS for Roaming Activation Procedure. If the NS is neither the Home NS of the End-Device nor configured to use the JS, then the NS SHALL silently ignore the Join-request and the procedure terminates here.

The NS SHALL use DNS to lookup the IP address of the JS based on the JoinEUI in the received Join-request message (see Section 19), if the NS is not already configured with the IP address/hostname of the JS by an out-of-band mechanism. If DNS lookup fails, then the NS SHALL terminate the procedure here.

For R1.0 [LW10] End-Devices configured with an AppEUI not identifying a Join Server, the NS SHOULD be configured with the IP address/hostname of the JS by an out-of-band mechanism.

Step 3:

The NS sends a JoinReq message to the JS carrying the PHYPayload of the Join-request message, MACVersion, DevEUI, DevAddr, DLSettings, RxDelay, and optionally CFList. The NS SHALL set the value of the MACVersion to the highest common version between the End-Device and the NS.

Step 4:

The JS SHALL process the Join-request message according to the MACVersion and send JoinAns to the NS carrying Result=Success, PHYPayload with Join-accept message, network session keys (SNwkSIntKey, FNwkSIntKey, and NwkSEncKey in case of a R1.1, and NwkSKey in case of a R1.0/1.0.2 End-Device), either the encrypted AppSKey or SessionKeyID or both, and Lifetime in case of success, and Result=UnknownDevEUI or MICFailed in case of failure (e.g., if the End-Device is not recognized by the JS, or if the MIC of the Join-request fails the verification).

JS may create SessionKeyID which is associated with the generated session keys.

SNwkSIntKey, FNwkSIntKey, NwkSEncKey, and AppSKey are generated based on the LoRaWAN 1.1 specification [LW11] for R1.1 End-Devices. NwkSKey is generated based on the LoRaWAN 1.0 specification [LW10] for R1.0/R1.0.2 End-Devices. AppSKey is encrypted using a key shared between the JS and the AS when it is delivered from the JS to the NS.

For R1.0 [LW10] End-Devices, the JS SHALL process the Join-request message also when the AppEUI is not identifying the JS.

Step 5:

The NS SHALL forward the received PHYPayload with Join-accept message to the End-Device if the received JoinAns message indicates Success. The End-Device SHALL generate the network session keys, and AppSKey based on the LoRaWAN specification [LW10, LW102, LW11] upon receiving the Join-accept message.

Step 6:

When the NS receives an uplink packet from the End-Device, the NS SHALL send the encrypted AppSKey or SessionKeyID or both along with the application payload to the AS.

Step 7:

When AS receives the encrypted AppSKey along with the application payload, then the AS SHALL decrypt the AppSKey using a secret key shared between the JS and the AS, and use the AppSKey to decrypt the received payload. If the encrypted AppSKey is not made available by the NS, then the AS SHALL proceed to the next step.

Step 8:

This step takes place in case the AS wants to receive the AppSKey directly from the JS.

The AS SHALL request the AppSKey identified by the DevEUI of the End-Device and the SessionKeyID from the JS. The AppSKey is encrypted using a shared secret between the JS and the AS. The JS sends the encrypted AppSKey, DevEUI and the SessionKeyID to the AS. Then the AS SHALL decrypt the encrypted AppSKey using a secret key shared between the JS and the AS. Then, the AS starts using the AppSKey to encrypt and decrypt the application payload.

OTA activation of a commissioned End-Device can happen both when the NS and the JS belong to the same administrative domain and when they belong to two separate administrative domains.

9 Deactivation (Exit) of OTA End-Devices

LoRa session of an OTA-activated End-Device can also be terminated for various reasons, such as user reaching end of contract, malicious End-Device behavior, etc. The procedure used for deactivating the session is the Exit Procedure, which is the counter-part of the Join Procedure.

There is no explicit and dedicated LoRaWAN signaling for performing the Exit Procedure. It is assumed that the End-Device and the backend rely on application-layer signaling to perform this procedure. Triggers and the details of application-layer signaling are outside the scope of this specification.

When the hNS is notified about the Exit Procedure by the AS and there is a separate sNS, then the hNS SHALL perform Handover Roaming Stop Procedure to convey the termination of the LoRaWAN session to the sNS.

The End-Device successfully performing a new Join/Rejoin Procedure also terminates the current LoRaWAN session, and in a way, it can be considered as the Deactivation associated with that session.

10 Security Associations

Table 1 shows the security associations used by the LoRaWAN deployments. Some of the required security associations will be detailed in the LoRaWAN specification, and some are left to the deployments.

End-points	Type	In or out of scope for LoRa spec.	Used for	Created by (if dynamic)	Key names
ED-JS	Static	In-scope	Securing Join/Rejoin	-	AppKey, NwkKey
ED-NS	Dynamic	In-scope	Securing over-the-air frame delivery	Join Procedure	SNwkSIntKey, FNwkSIntKey, NwkSEncKey, NwkSKey
ED-AS	Dynamic	In scope	Securing end-to-end frame payload delivery	Join Procedure	AppSKey
JS-NS	Static	Out of scope	Securing Join/Rejoin and session key delivery	-	-
AS-JS	Static	In scope	Securing AppSKey delivery	-	ASJSKey
	Static	Out of scope	Commissioning/ Decommissioning	-	ASJSKey
JS-Manufacturer	Static	Out of scope	Securing AppKey/NwkKey delivery	-	-
AS-NS	Static	Out of scope	Securing frame delivery	-	-
NS-NS	Static	Out of scope	Securing Join/Rejoin and inter-NS frame delivery	-	-

Table 1 LoRaWAN security associations

11 Roaming Procedure

11.1 Types of Roaming

There are two types of LoRa roaming, namely Passive Roaming and Handover Roaming. Passive Roaming enables the End-Device to benefit from LoRaWAN service of a Network Server (NS) even when the End-Device is using the Gateway(s) (GWs) under the control of another NS, within the limits of the overlapping RF capabilities (i.e., channels) of the two networks, for that End-Device. LoRa Session and the MAC-layer control of the End-Device are maintained by the former NS, which is called the Serving NS (sNS), whereas the frame forwarding to/from air interface is handled by the latter NS, which is called the Forwarding NS (fNS). There can only be one sNS for a given LoRa Session whereas zero or more fNSs may be involved with the same session.

There are two types of fNSs: Stateful and stateless. A stateful fNS creates context at the onset of the passive roaming of an End-Device and utilizes that context for processing any subsequent uplink/downlink packets of the same End-Device. A stateless fNS does not create any such context and therefore ends up having to process any uplink/downlink packet independent of each other. It is assumed that whether a given fNS is stateless or stateful is known to its roaming partners by some out of scope mechanism.

Handover Roaming enables the transfer of the MAC-layer control from one NS to another. hNS maintains the control-plane and data-plane with the JS and the AS even after the End-Device performs a Handover Roaming from one NS to another. hNS stays the same for a given LoRa Session until the End-Device performs the next Join Procedure. Unlike the fNS, the sNS has capability to control the End-Device RF settings, which allows more flexible roaming scenarios.

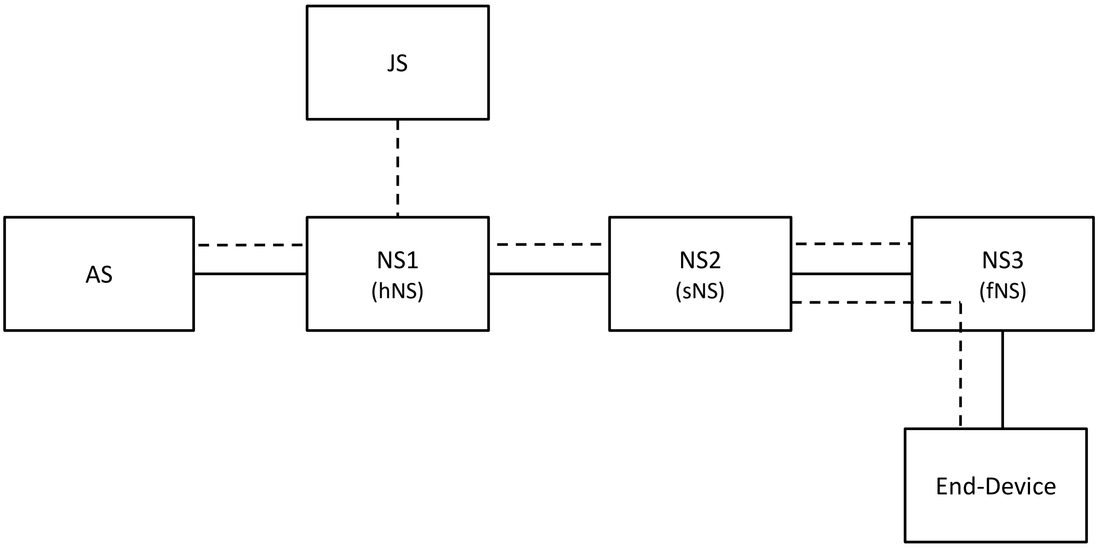

Figure 6 Use of Handover and Passive Roaming

Figure 6 illustrates an example case where both the Handover Roaming and Passive Roaming are used for a LoRa Session simultaneously. In this example, the End-Device was activated through NS1 which acts as the hNS. Subsequently, the End-Device has performed Handover Roaming from NS1 to NS2 when NS2 became the sNS, and also Passive Roaming from NS2 to NS3 when NS3 became the fNS for the End-Device.

Roaming activation is the capability for an End-Device to activate under the coverage of a Visited NS.

This specification describes the procedures for the following roaming cases:

- Passive Roaming during an ongoing LoRa Session
- Handover Roaming during an ongoing LoRa Session
- Roaming Activation of a new LoRa Session based on Handover Roaming between the Home NS and the Visited NS
- Roaming Activation of a new LoRa Session based on Passive Roaming between the Home NS and the Visited NS

Activation of a new LoRa Session when the Home NS and the Visited NS do not have any roaming agreement is outside the scope of this specification. This includes the case where the two NSs may have roaming agreement with a third NS (e.g., Home NS and 3rd NS having a Handover Roaming agreement, and the 3rd NS and the Visited NS having a Passive Roaming agreement).

11.2 Roaming Policy

Each network operator SHALL be configured with a roaming policy that can individually allow/disallow Passive Roaming, Handover Roaming, Passive Roaming based Activation, Handover Roaming based Activation with other network operators identified by their NetIDs. For Passive Roaming, the policy SHALL also include whether the uplink MIC check is done by the fNS or not.

Each network operator SHALL be configured with a roaming policy that can allow/disallow Passive Roaming, Handover Roaming, Passive Roaming based Activation, Handover Roaming based Activation of its individual End-Devices identified by the DevEUI.

11.3 Passive Roaming

This procedure applies to both R1.0 [LW10, LW102] and R1.1 [LW11] End-Devices and networks.

11.3.1 Passive Roaming Start

Figure 7 illustrates the message flow for Passive Roaming Procedure between two NSs for an ongoing LoRa Session of an End-Device. Refer to Section 12.2 for Passive Roaming based Activation of a new LoRa Session.

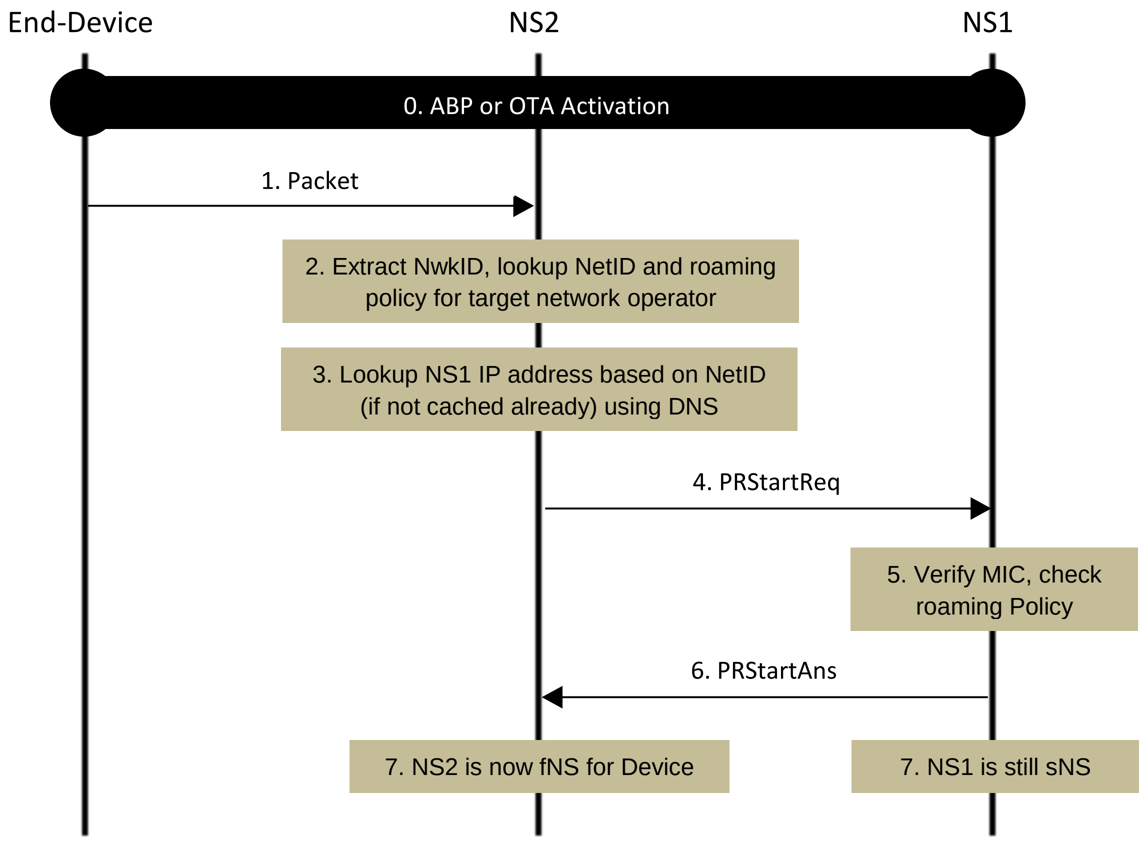

Figure 7 Passive Roaming start

Step 0:

The End-Device is activated through the NS1.

Step 1:

When the End-Device transmits a packet, it is received by the NS2 which does not have any context associated with the End-Device.

Step 2:

If the NS2 is configured to enable passive roaming with other network operators, then the NS2 SHALL attempt to map the NwkID in the received packet with the NetID(s) of the operators with whom it has a passive roaming agreement. If no match is found, then the NS2 SHALL discard the packet and the procedure terminates here.

Step 3:

If one or more matching NetIDs are found, then the NS2 SHALL use DNS to lookup (see Section 19) the IP address of NS for each matching NetID (e.g., NS1 in this case), if the NS2 is not already configured with the IP address/hostname of the NS by an out-of-band mechanism. If there are more than one match, then Steps 4-6 are executed for each matching NS.

Step 4:

The NS2 SHALL send a PRStartReq message to the NS1, carrying the PHYPayload of the incoming packet, and the associated ULMetadata. Details of metadata are described in Section 16.

Step 5:

The NS1 SHALL check if it already has a passive roaming agreement with the network operator identified by the received NetID, and decide to return a PRStartAns carrying Result=NoRoamingAgreement if no agreement is found.

The NS1 SHALL extract the DevAddr of the End-Device from the PHYPayload, identify the corresponding network session integrity key (SNwkSIntKey and FNwkSIntKey in case of R1.1, and NwkSKey in case of R1.0/1.0.2 End-Device), and verify the MIC in the PHYPayload. If the keys are not found or if the MIC verification fails, then the NS1 SHALL decide to return a PRStartAns carrying Result=MICFailed.

Step 6:

If the identified End-Device is configured to use Passive Roaming and the NS1 decides to enable Passive Roaming via the NS2, then the NS1 SHALL send a PRStartAns to the NS2 carrying the Result=Success, and Lifetime associated with the Passive Roaming. The NS1 SHALL also include DevEUI and ServiceProfile if NS2 is to operate as a stateful fNS, and FCntUp and FNwkSIntKey (in case of R1.1) or NwkSKey (in case of R1.0/1.02) in the PRStartAns message if NS1-NS2 Passive Roaming agreement requires the NS2 to perform MIC check on the uplink packets.

If the NS1 does not wish to enable Passive Roaming via NS2 at this point in time, then it SHALL send a PRStartAns to the NS2 carrying Result=Deferred, and Lifetime. The NS2 SHALL not send any more PRStartReq to the NS1 for the same End-Device for the duration of Lifetime upon receiving this message.

If a failure has occurred at Step 5, then the NS1 SHALL send a PRStartAns to the NS2 carrying the identified Result.

The NS1 may receive PRStartReq from multiple NSs at the same time, and decide to enable Passive Roaming with zero or more of them.

The NS1 and the NS2 SHALL terminate the Passive Roaming on their own (i.e., without involving additional signaling with each other) after the associated Lifetime expires, unless the Passive Roaming is extended with a new round of PRStartReq/PRStartAns before the expiration. For stateless fNS operation, the NS1 SHALL set the value of Lifetime associated with the Passive Roaming to zero.

Step 7:

The NS2 becomes an fNS for the LoRa Session of the End-Device as soon as it receives the successful PRStartAns. NS1 continues to serve as the sNS.

After this point on, the NS2 SHALL forward packets received from the End-Device to the NS1, and the NS1 SHALL accept such packets from NS2. Also, the NS1 SHALL note the NS2 as a candidate fNS for sending packets to the End-Device. The NS2 SHALL accept packets sent from NS1 to be forwarded to the End-Device via one of its GWs.

11.3.2 Packet Transmission

Figure 8 illustrates the message flow for an End-Device sending and receiving packets using Passive Roaming. Even though the flow shows an uplink packet immediately followed by a downlink packet, the uplink and the downlink parts of the flow can be executed independently in any order as allowed by the class of the End-Device.

In case of stateless fNS procedure, each uplink packet SHALL be processed according to Section 11.3.1 (not according to the Steps 1-4 in this section, which assume stateful fNS). Nevertheless, Steps 5-11 in this section are applicable to downlink packet processing even for stateless fNS procedure.

All steps in this section are applicable to uplink and downlink packet processing in case of stateful fNS procedure.

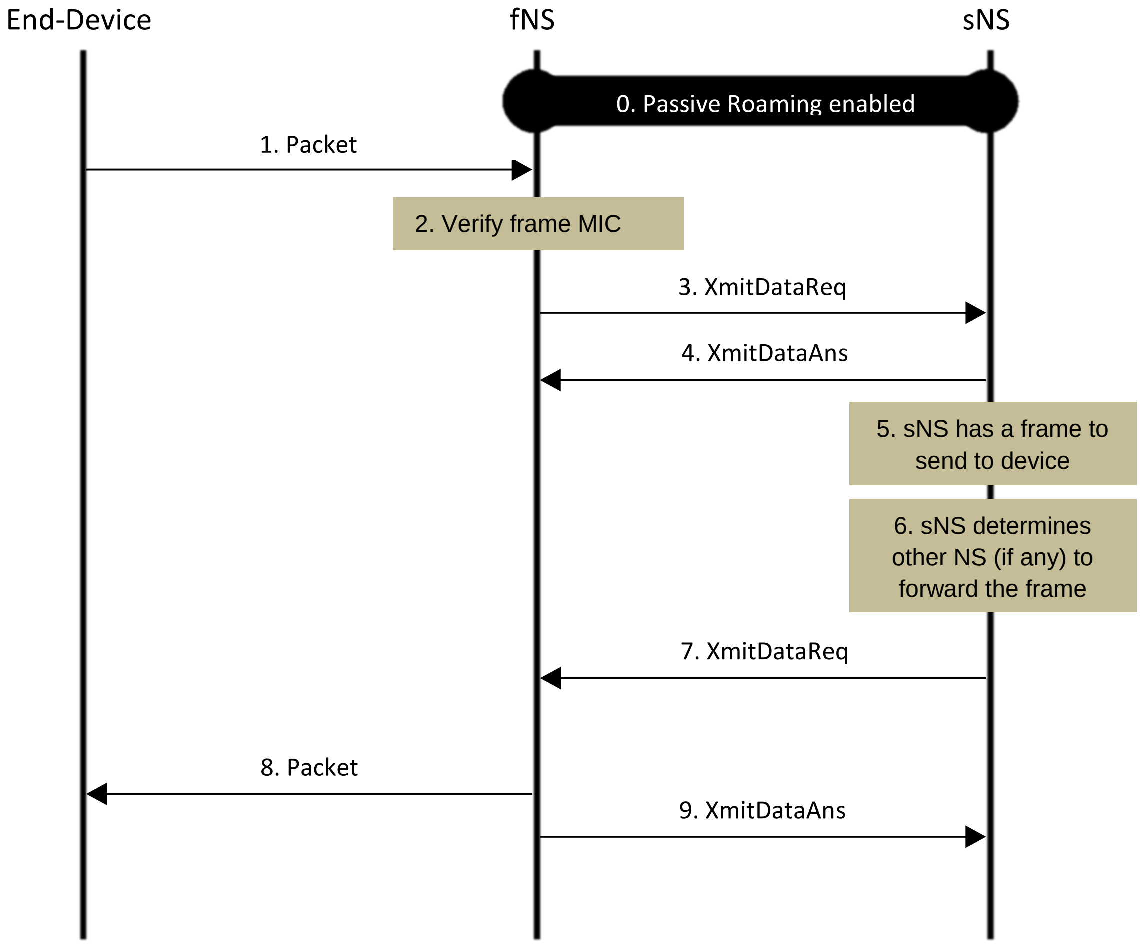

Figure 8 Packet transmission using Passive Roaming

Step 0:

Stateful Passive Roaming is enabled between the fNS and the sNS for the End-Device.

Step 1:

The End-Device transmits a packet, which is received by the fNS.

Step 2:

If the fNS is required to perform MIC check on the uplink packets based on sNS-fNS Passive Roaming agreement, then the fNS SHALL extract the DevAddr of the End-Device from the packet and identify the FNwkSIntKey/NwkSKey, and verify the MIC in the packet. If no FNwkSIntKey/NwkSKey is found or if the MIC verification fails, then the fNS SHALL discard the packet.

Step3:

If an End-Device is identified, the fNS SHALL send a XmitDataReq message to the identified End-Device's sNS carrying the PHYPayload of the received packet and the associated ULMetadata.

Step 4:

The sNS SHALL send a XmitDataAns message back to the fNS carrying Result upon receiving the XmitDataReq.

The subsequent steps are executed when the sNS has a packet to send to the End-Device, which may or may not follow the preceding steps.

Step 5:

The sNS has a packet to send to the End-Device.

Step 6:

The sNS SHALL determine whether to send the packet via one of the GWs under its control or via a GW under the control of an fNS.

Step 7:

If the sNS decides to send the packet via an fNS, the sNS SHALL send XmitDataReq message to the fNS carrying the PHYPayload of the packet, and DLMetadata.

Step 8:

If there is an error condition in the received XmitDataReq, the fNS SHALL send a XmitDataAns message to the sNS carrying Result set to a failure value and SHALL NOT attempt to transmit the packet. Otherwise, the fNS SHALL attempt to transmit the packet to the End-Device based on the metadata information it has received from the sNS. If the metadata includes GWInfo.ULToken, then the fNS may use that for selecting the downlink transmission GW. The fNS may fail to transmit the packet due to the timing constraints and the network conditions. In that case, the fNS SHALL not retry transmission.

Step 9:

After attempting to transmit the packet, the fNS SHALL send a XmitDataAns message to the sNS carrying one or both of DLFreq1 and DLFreq2 (depending on whether the packet was transmitted at RX1 or RX2 or both) with Result=Success for successful transmission, and Result=XmitFailed value otherwise.

11.3.3 Passive Roaming Stop

Figure 9 and Figure 10 illustrate the message flows for terminating Passive Roaming. This procedure is applicable to only stateful fNS.

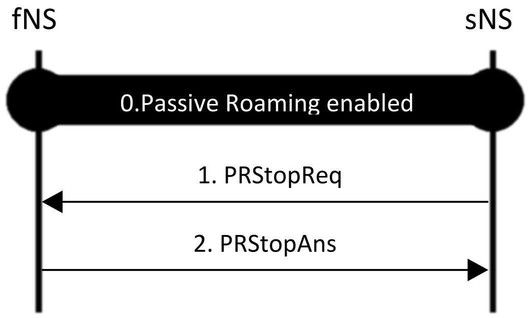

Figure 9 sNS-initiated Passive Roaming termination

Step 0:

Passive Roaming is enabled between the fNS and the sNS for the End-Device.

Step 1:

When sNS decides to terminate Passive Roaming for the End-Device before the expiration of the Passive Roaming lifetime, the sNS SHALL send a PRStopReq message to the fNS carrying the DevAddr and DevEUI of the End-Device, and optionally Lifetime. The sNS SHALL include Lifetime if the fNS is stateful and the sNS does not wish to receive another PRStartReq from the fNS for the End-Device within the stated time span.

Step 2:

The fNS SHALL verify that the End-Device with DevEUI is already in Passive Roaming and associated with the sNS. If both conditions are satisfied, then the fNS SHALL send PRStopAns message to the sNS carrying Result=Success. Otherwise, the fNS SHALL send PRStopAns message to the sNS carrying Result=UnknownDevEUI. If the received PRStopReq message included Lifetime, then the fNS SHALL not send another PRStartReq to the sNS for the End-Device until the Lifetime expires.

In case Passive Roaming for the End-Device was previously terminated with a PRStopReq message or refused with PRStartAns/Result=Deferred, a new PRStopReq message with a 0 value for Lifetime enables NS2 to send again PRStartReq for the End-Device as soon as it receives a packet from that End-Device. This applies only for stateful fNS.

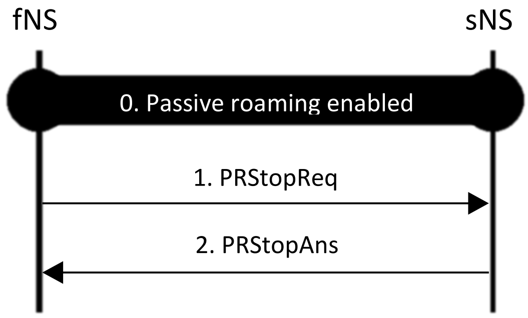

Figure 10 fNS-initiated Passive Roaming termination

Step 0:

Passive Roaming is enabled between the fNS and the sNS for the End-Device.

Step 1:

When the fNS decides to terminate Passive Roaming for the End-Device before the expiration of the Passive Roaming lifetime, the fNS SHALL send PRStopReq message to the sNS carrying the DevEUI of the End-Device.

Step 2:

The sNS SHALL verify that the End-Device with DevEUI is served by itself and it is already in Passive Roaming with the fNS. If both conditions are satisfied, then the sNS SHALL send PRStopAns message to the fNS carrying Result=Success. Otherwise, the sNS SHALL send PRStopAns message to the fNS carrying Result=UnknownDevEUI.

After the Passive Roaming terminates, the sNS and the fNS SHALL stop forwarding packets towards each other for the designated End-Device.

11.4 Handover Roaming

This procedure applies to only R1.1 [LW11] End-Devices and networks.

11.4.1 Handover Roaming Start

Figure 11 illustrates the message flow for Handover Roaming Procedure for an ongoing LoRa Session of an End-Device. Refer to Section 12.1 for Handover Roaming based Activation of a new LoRa Session.

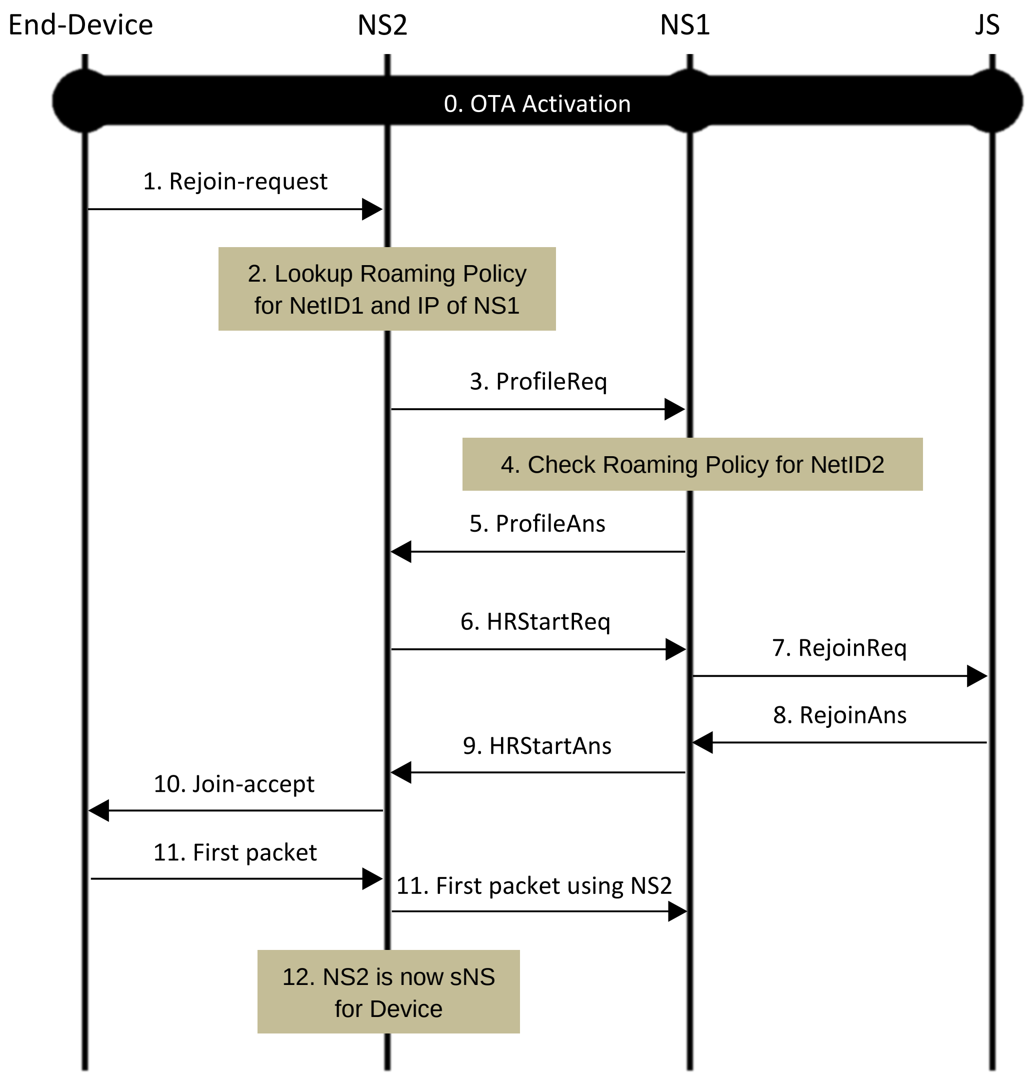

Figure 11 Handover Roaming start

Step 0:

Consider the case the End-Device has performed Activation on the NS1. Therefore, NS1 is acting as the fNS, sNS and hNS for the End-Device.

Step 1:

The End-Device transmits a Rejoin-request Type 0 message either in response to receiving a ForceRejoinReq MAC command (not shown) or autonomously without an external trigger.

Step 2:

If the NS2 is acting as the sNS for the End-Device as identified by the received DevEUI and the NS2 has not requested a Rejoin-request Type 0 by sending a ForceRejoinReq, then the NS2 SHOULD silently drop the message and the procedure terminates here.

If the NS2 is not the sNS for the End-Device, then the NS2 SHALL lookup its roaming policy with the operator identified by the NetID in the Rejoin-request. If the NS2 is not configured to enable Handover Roaming with the identified operator, then the NS2 SHALL discard the Rejoin-request and the procedure terminates here. Otherwise, the NS2 SHALL discover the IP address of the NS1 using DNS (see Section 19), if the NS2 is not already configured with the IP address/hostname of the NS1 by an out-of-band mechanism.

Step 3:

The NS2 SHALL send an ProfileReq message to the NS1 carrying DevEUI if the NS2 does not have the Device Profile of the End-Device in its cache. Steps 4 and 5 are skipped if the ProfileReq is not sent.

Step 4:

The NS1 SHALL lookup its roaming policy with the operator identified by the received NetID.

Step 5:

The NS1 SHALL send an ProfileAns to the NS2 carrying Result=Success, Device Profile, and Device Profile Timestamp (which carries the timestamp of the last Device Profile change) if the NS1 is configured to enable Handover Roaming with the NS2 and for the End-Device. If Handover Roaming is not allowed, then the ProfileAns carries Result=NoRoamingAgreement or DevRoamingDisallowed, and Lifetime, and the procedure terminates here. The Lifetime allows the NS1 to request the NS2 not to send additional ProfileReq for the End-Device until the Lifetime expires.

Step 6:

The NS2 SHALL send a HRStartReq message to the NS1 carrying the PHYPayload with Rejoin-request message, MACVersion, ULMetadata, Device Profile Timestamp, and the parameters DevAddr, DLSettings, RxDelay, and optionally CFList identified by the NS2 to be assigned to the End-Device. The NS2 SHALL set the value of the MACVersion to the highest common version between the End-Device and the NS2.

Step 7:

If Handover Roaming is not allowed with the NS2 or for the End-Device, or if the MIC verification of the message has failed, then the NS1 SHALL proceed to Step 9. Handover Roaming rejection may be due to the per-NS or per-device roaming policy, or potential unnecessity of Handover Roaming while the End-Device is already being served by another sNS.

If the NS1 determines that the Device Profile has changed since the time indicated by the received Device Profile Timestamp, then the NS1 concludes that the NS2 has a stale Device Profile information. In that case, the NS1 SHALL proceed to Step 9.

Otherwise, the NS1 SHALL forward the RejoinReq message to the JS, carrying the PHYPayload with Rejoin-request message, MACVersion, DevEUI, DevAddr, DLSettings, RxDelay, and CFList as received from the NS2.

Step 8:

The JS SHALL process the Rejoin-request according to the MACVersion and send a RejoinAns message to the NS1 carrying Result=Success, the PHYPayload with Join-accept message, SNwkSIntKey, FNwkSIntKey, NwkSEncKey, Lifetime if the Rejoin-Request is accepted by the JS. Otherwise, the JS SHALL send a RejoinAns to the NS1 carrying Result=UnknownDevEUI or MICFailed.

The NS1 SHALL treat the received Lifetime value as the upper-bound of the session lifetime it assigns to the LoRa session.

Step 9:

If the NS1 decided not to allow Handover Roaming at Step 7, then the NS1 SHALL send a HRStartAns message to the NS2 carrying Result set to a failure value (see Table 24), and Lifetime. The Lifetime allows the NS1 to request the NS2 not to send additional HRStartReq for the End-Device until the Lifetime expires.

If the NS1 concluded that the Device Profile known to the NS2 is stale, then the NS1 SHALL send HRStartAns message to the NS2 carrying Result=StaleDeviceProfile, latest Device Profile, and its Device Profile Timestamp. The NS2 SHALL jump back to Step 6 to use the new Device Profile it just received.

Otherwise, the NS1 SHALL forward the payload of the received RejoinAns message in an HRStartAns message to the NS2 by also including DLMetadata and Service Profile. The NS1 SHALL also cache the received SNwkSIntKey, so that it can verify the MIC of the subsequent Rejoin-Type 0 messages before deciding to forward them to the JS.

Step 10:

If the HRStartAns message indicates Success, then the NS2 SHALL forward the received PHYPayload with Join-accept message to the End-Device. Otherwise, the NS2 SHALL not send any response back to the End-Device.

If the Rejoin Procedure was successful, then the NS2 SHALL start forwarding packets received from the End-Device to the NS1, and the NS1 SHALL accept such packets from the NS2. Also, the NS1 SHALL start forwarding packets received from the AS to the NS2, and the NS2 SHALL accept such packets from the NS1.

Step 11:

The End-Device sends its first uplink packet. The NS2 SHALL transmit that packet to the NS1.

Step 12:

The NS2 starts serving as the sNS and the NS1 stops serving as the sNS as soon as the first uplink packet is received from the End-Device. Meanwhile, the NS1 continues to serve as the hNS of the End-Device.

11.4.2 Packet Transmission

In case of Handover Roaming, the hNS and the sNS SHALL use XmitDataReq/Ans messages the same way they are used with the Passive Roaming (see Section 11.3.2). The only difference is, the hNS-sNS interface carries the FRMPayload instead of the PHYPayload, and the ULMetadata/DLMetadata includes different set of objects as described in Section 16.

11.4.3 Handover Roaming Stop

Figure 12 illustrates the hNS terminating the Handover Roaming with the previously serving sNS after the End-Device performs Handover Roaming to a new sNS.

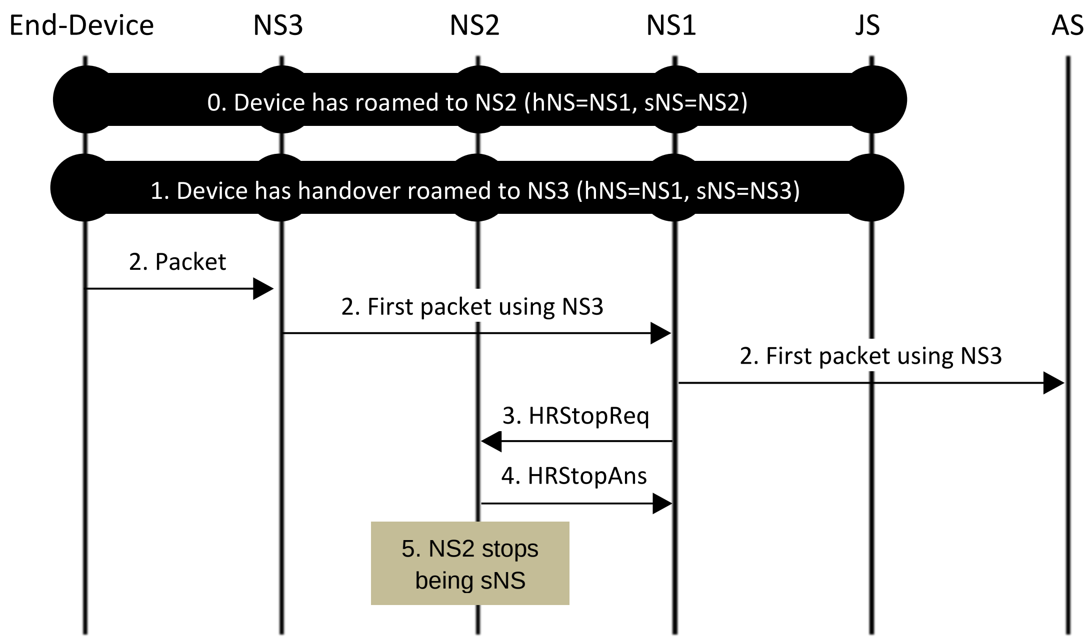

Figure 12 Termination of sNS

Step 0:

The End-Device performs Handover Roaming between the NS1 and the NS2.

Step 1:

The End-Device performs Handover Roaming between the NS1 and the NS3.

Step 2:

The very first uplink packet is received from the End-Device by the NS1 via the new sNS (NS3).

Step 3:

The NS1 SHALL send an HRStopReq message to the previously serving sNS (NS2) carrying DevEUI when it receives the first packet from the End-Device via the new sNS (NS3).

HRStopReq message carries optionally Lifetime, which means NS1 does not wish to receive another HRStartReq from NS2 for this DevEUI within the stated time span.

Step 4:

The previously serving sNS (NS2) SHALL terminate Handover Roaming and send an HRStopAns to the NS1 carrying Result=Success if the NS2 has active Handover Roaming for the End-Device identified with the received DevEUI and associated with the NS1. If the NS2 does not have an active Handover Roaming for the End-Device associated with the NS1, then the NS2 SHALL send an HRStopAns to the NS1 carrying Result=UnknownDevEUI.

Step 5:

The NS2 stops serving as the sNS for the LoRa session of the End-Device. If the NS2 has enabled Passive Roaming with another NS for the LoRa session of the End-Device, then the NS2 SHALL also terminate the Passive Roaming with that NS.

In case Handover Roaming for the End-Device was previously terminated with a HRStopReq command, a new HRStopReq command with a 0 value for Lifetime enables NS2 to send again HRStart requests for this End-Device as soon as it receives a new Rejoin-request Type 0 message.

Another case of Handover Roaming termination is when the sNS decides to terminate roaming. The sNS may precede the termination procedure by sending a ForceRejoinReq command to the End-Device. Then, the sNS SHALL send an HRStopReq to the hNS carrying the DevEUI. The hNS SHALL terminate Handover Roaming and send an HRStopAns to the sNS carrying Result=Success if the hNS has active Handover Roaming for the End-Device identified with the received DevEUI and associated with the sNS. If the hNS does not have an active Handover Roaming for the End-Device associated with the sNS, then the hNS SHALL send an HRStopAns to the sNS carrying Result=UnknownDevEUI. The sNS may still terminate the Handover Roaming even if it received a failure Result from the hNS.

11.4.4 Home NS Regaining Control

Figure 13 illustrates the message flow of the hNS becoming the sNS by taking the control from currently serving sNS.

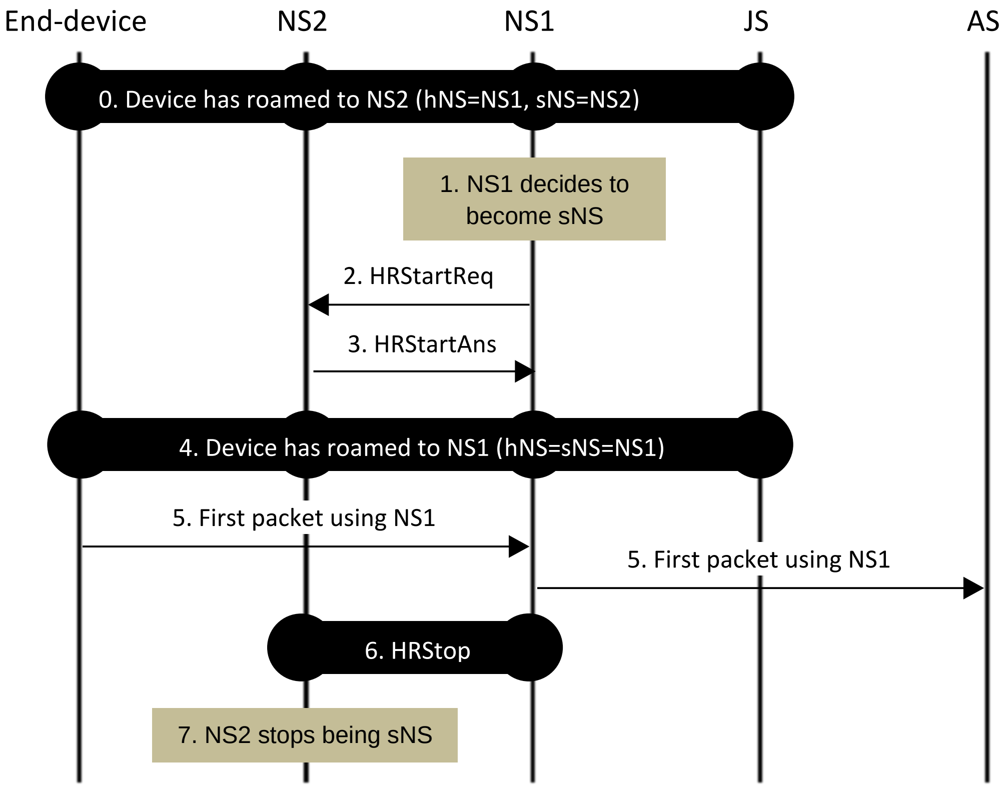

Figure 13 hNS regaining sNS control

Step 0:

The End-Device performs Handover Roaming between the NS1 and the NS2.

Step 1:

The NS1 decides to become the sNS.

Step 2:

The NS1 SHALL send an HRStartReq message to the NS1 carrying DevEUI to trigger the Handover Roaming.

Step 3:

The NS2 SHALL send HRStartAns to the NS1 carrying Result=Success if the NS2 has active Handover Roaming for the End-Device identified by the received DevEUI and associated with the NS1, Result=UnknownDevEUI otherwise.

Step 4:

The NS2 SHALL initiate network-triggered Handover Roaming as described in Section 11.4.1. It is assumed that the End-Device is within the radio coverage of the NS1 when this procedure is initiated, and the NS1 rejects Handover Roaming attempt from other NSs, including NS2, and becomes the sNS.

Step 5:

The very first uplink packet is received from the End-Device directly by the NS1.

Step 6:

The NS1 SHALL perform Handover Roaming Stop Procedure with the NS2 as described in Section 11.4.3.

Step 7:

The NS2 stops serving as the sNS for the LoRa Session of the End-Device. If the NS2 has enabled Passive Roaming with another NS for the LoRa session of the End-Device, then the NS2 SHALL also terminate the Passive Roaming with that NS.

Alternatively, the NS1 can wait until the End-Device decides to initiate Handover Roaming on its own, effectively skipping the Steps 2 and 3, and continuing with the Steps 4-7.

12 OTA Roaming Activation Procedure

This section describes the procedures for activation of a new LoRa Session when the End-Device is outside the coverage of its Home NS but under the coverage of a Visited NS.

It is assumed that the Home NS is aware of the roaming capabilities of the Visited NS, and the Home NS decides which type of activation (Passive Roaming or Handover Roaming based) will be performed.

12.1 Handover Roaming Activation

This procedure applies to both R1.0 [LW10, LW102] and R1.1 [LW11] End-Devices and networks.

12.1.1 Handover Roaming Start

Figure 14 illustrates the message flow for OTA Handover Roaming Activation Procedure.

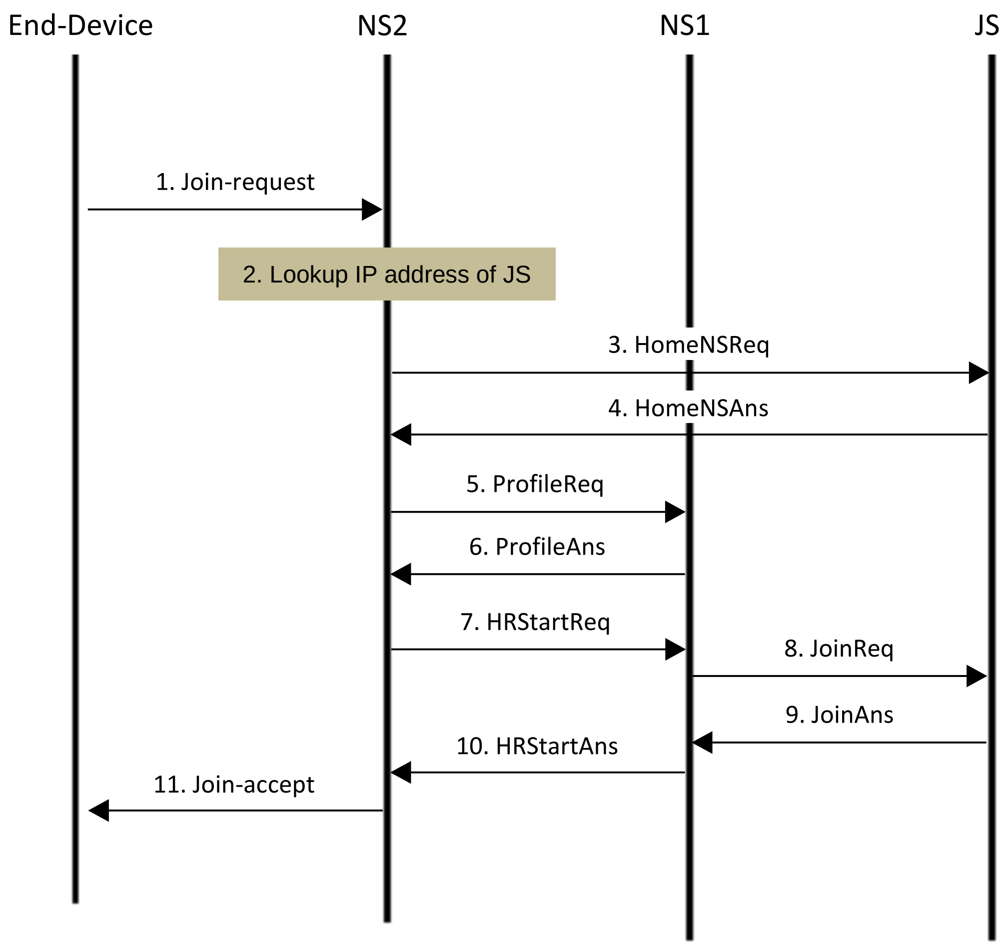

Figure 14 Message flow for Handover Roaming Activation Procedure.

Step 1:

The End-Device SHALL transmit a Join-request message.

Step 2:

When the NS2 receives the Join-request message, the NS2 SHALL determine whether it is the Home NS for the End-Device identified by DevEUI, or not. In this flow, it is assumed that the NS2 is not the Home NS of the End-Device. See Section 8 for the other case.

The NS2 SHALL determine whether it is configured to work with the JS identified by the JoinEUI or not. If it is not configured so, then the NS2 SHALL terminate the procedure here.

The NS2 SHALL use DNS to lookup the IP address of the JS based on the JoinEUI in the received Join-request message (see Section 19), if the NS2 is not already configured with the IP address/hostname of the JS by an out-of-band mechanism. If DNS lookup fails, then the NS2 SHALL terminate the procedure here.

Step 3:

If the NS2 already knows the identity of the Home NS of the End-Device, then Steps 3 and 4 are skipped. Otherwise, the NS2 SHALL send an HomeNSReq message to the JS carrying the DevEUI of the Join-request message.

Step 4:

The JS SHALL send an HomeNSAns message to the NS2 carrying Result=NoRoamingAgreement if the NS2 is not in the authorized networks as listed in the JS to serve the End-Device for Roaming Activation, and the procedure terminates here.

The JS SHALL send HomeNSAns message to the NS2 carrying Result=Success and HNetID of the End-Device (NetID of NS1).

Step 5:

If the NS2 already knows the Device Profile of the End-Device, and NS2 only has Handover Roaming agreement with NS1, then Steps 5 and 6 are skipped. Otherwise, the NS2 SHALL use DNS to lookup the IP address of the NS1 based on the NetID in the received Join-request message (see Section 19), if the NS2 is not already configured with the IP address/hostname of the NS1 by an out-of-band mechanism. If DNS lookup fails, then the NS2 SHALL terminate the procedure here.

The NS2 SHALL send a ProfileReq message to the NS1 carrying the DevEUI.

Step 6:

If there is no business agreement between the NS1 and the NS2, then the NS1 SHALL send an ProfileAns message to the NS2 carrying Result=NoRoamingAgreement. If the NS1 could not identify the End-Device with the DevEUI, then the NS1 SHALL send a ProfileAns message to the NS2 carrying Result=UnknownDevEUI. If the End-Device is not allowed to perform Roaming Activation, then the NS1 SHALL send a ProfileAns message to the NS2 carrying Result=RoamingActDisallowed. Otherwise, assuming the NS1 decides to enable Handover Roaming Activation, the NS1 SHALL send a ProfileAns message to the NS2 carrying Result=Success, RoamingActivationType=Handover, Device Profile, and Device Profile Timestamp (which carries the timestamp of the last Device Profile change).

The following steps describe the procedure when the RoamingActivationType is Handover.

Step 7:

If the Result of incoming ProfileAns indicates Success, or if the Steps 5 and 6 are skipped, then the NS2 SHALL send an HRStartReq message to the NS1 carrying the PHYPayload with Join-Request message, MACVersion, ULMetadata, DevAddr, DLSettings, RxDelay, optionally CFList, and Device Profile Timestamp. The NS2 SHALL set the value of the MACVersion to the highest common version between the End-Device and the NS2.

Step 8:

When steps 5 and 6 are skipped, if there is no business agreement between the NS1 and the NS2 or if the NS1 could not identify the End-Device with the DevEUI or if the End-Device is not allowed to perform Roaming Activation then the NS1 shall proceed to Step 10.

If the NS1 determines that the Device Profile has changed since the time indicated by the received Device Profile Timestamp, then the NS1 concludes that the NS2 has a stale Device Profile information. In that case, the NS1 SHALL proceed to Step 10. Otherwise, the NS1 sends a JoinReq message to the JS carrying the PHYPayload with Join-request message, MACVersion, DevEUI, DevAddr, DLSettings, RxDelay, and CFList as received from the NS2.

Step 9:

The JS SHALL process the Join-request message according to the MACVersion and send JoinAns to the NS1 carrying Result=Success, PHYPayload with Join-accept message, network session keys (SNwkSIntKey, FNwkSIntKey, and NwkSEncKey in case of a R1.1, and NwkSKey in case of a R1.0/1.0.2 End-Device), encrypted AppSKey or SessionKeyID or both, Lifetime in case of success, and Result=UnknownDevEUI or MICFailed in case of failure (e.g., if the End-Device is not recognized by the JS, or if the MIC of the Join-request fails the verification). Network session keys, and AppSKey are generated based on the LoRaWAN specification [LW10, LW11]. AppSKey is encrypted using a key shared between the JS and the AS when it is delivered from the JS to the NS.

Step 10:

If there is no business agreement between the NS1 and the NS2, then the NS1 SHALL send an HRStartAns message to the NS2 carrying Result=NoRoamingAgreement. If the NS1 could not identify the End-Device with the DevEUI, then the NS1 SHALL send a HRStartAns message to the NS2 carrying Result=UnknownDevEUI. If the End-Device is not allowed to perform Roaming Activation, then the NS1 SHALL send a HRStartAns message to the NS2 carrying Result= RoamingActDisallowed.

If the NS1 concluded that the Device Profile known to the NS2 is stale, then the NS1 SHALL send HRStartAns message to the NS2 carrying Result=StaleDeviceProfile, latest Device Profile, and its Device Profile Timestamp. In this case, the NS2 SHALL jump back to Step 7 to use the new Device Profile it just received.

Otherwise, the NS1 SHALL send an HRStartAns message to the NS2. The HRStartAns SHALL contain the same objects as the JoinAns message described in Step 9 and also the Service Profile of the End-Device.

In case of a R1.1 End-Device, the NS1 SHALL also cache the received SNwkSIntKey, so that it can verify the MIC of the subsequent Rejoin-Type 0 messages before deciding to forward them to the JS.

Step 11:

The NS2 SHALL forward the received PHYPayload with Join-accept message to the End-Device if HRStartAns message indicates success. The End-Device SHALL generate network session keys, and AppSKey based on the LoRaWAN specification [LW10, LW11] upon receiving the Join-accept message.

If encrypted AppSKey is not made available by the JS to the AS via the NS, then the AS SHALL retrieve it directly from the JS using the same method as defined in Step 8 of OTA Activation at Home Procedure (see Section 8).

12.1.2 Packet Transmission

The details of uplink and downlink packet transmission between the hNS and the sNS after the two are engaged in Roaming Activation for an End-Device are same as the Handover Roaming case as described in Section 11.4.2.

12.1.3 Handover Roaming Stop

Handover Roaming Stop Procedure (Section 11.4.3) is used when either the hNS or the sNS decides to terminate the roaming.

12.2 Passive Roaming Activation

This procedure applies to both R1.0 [LW10, LW102] and R1.1 [LW11] End-Devices and networks.

12.2.1 Passive Roaming Start

Figure 15 illustrates the message flow for OTA Passive Roaming Activation Procedure.

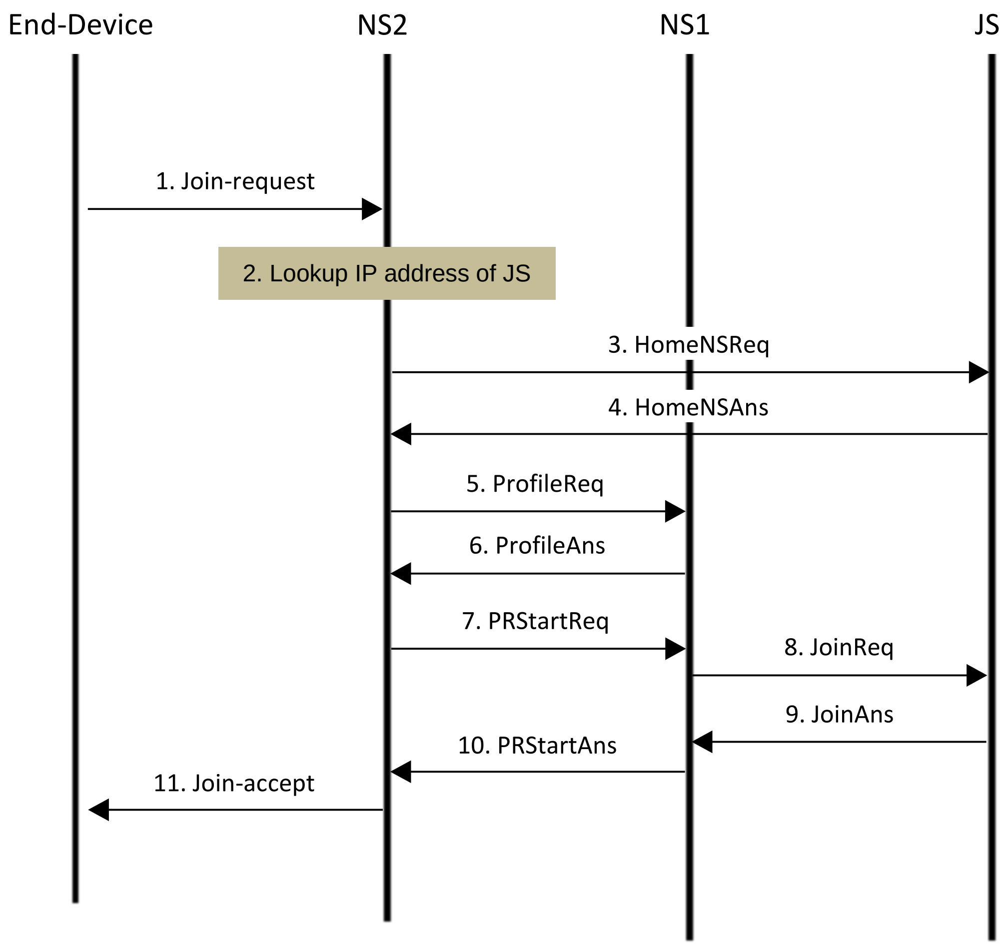

Figure 15 Message flow for Passive Roaming Activation Procedure.

Step 1:

The End-Device SHALL transmit a Join-request message.

Step 2:

When the NS2 receives the Join-request message, the NS2 SHALL determine whether it is the Home NS for the End-Device identified by DevEUI, or not. In this flow, it is assumed that the NS2 is not the Home NS of the End-Device. See Section 8 for the other case.

The NS2 SHALL determine whether it is configured to work with the JS identified by the JoinEUI or not. If it is not configured so, then the NS2 SHALL terminate the procedure here.

The NS2 SHALL use DNS to lookup the IP address of the JS based on the JoinEUI in the received Join-request message (see Section 19), if the NS2 is not already configured with the IP address/hostname of the JS by an out-of-band mechanism. If DNS lookup fails, then the NS2 SHALL terminate the procedure here.

Step 3:

If the NS2 already knows the identity of the Home NS of the End-Device, then Steps 3 and 4 are skipped. Otherwise, the NS2 SHALL send an HomeNSReq message to the JS carrying the DevEUI of the Join-request message.

Step 4:

The JS SHALL send an HomeNSAns message to the NS2 carrying Result=NoRoamingAgreement if the NS2 is not in the authorized networks as listed in the JS to serve the End-Device for Passive Roaming Activation, and the procedure terminates here.

The JS SHALL send HomeNSAns message to the NS2 carrying Result=Success and hNS of the End-Device (NetID of NS1).

Step 5:

If the NS2 only has Passive Roaming agreement with NS1, then Steps 5 and 6 are skipped. Otherwise, the NS2 SHALL use DNS to lookup the IP address of the NS1 based on the NetID received from the JS, if the NS2 is not already configured with the IP address/hostname of the NS1 by an out-of-band mechanism. If DNS lookup fails, then the NS2 SHALL terminate the procedure here.

The NS2 SHALL send a ProfileReq message to the NS1 carrying the DevEUI.

Step 6:

If there is no business agreement between the NS1 and the NS2, then the NS1 SHALL send an ProfileAns message to the NS2 carrying Result=NoRoamingAgreement. If the NS1 could not identify the End-Device with the DevEUI, then the NS1 SHALL send a ProfileAns message to the NS2 carrying Result=UnknownDevEUI. If the End-Device is not allowed to perform Roaming Activation, then the NS1 SHALL send a ProfileAns message to the NS2 carrying Result=RoamingActDisallowed. Otherwise, assuming the NS1 decides to enable Passive Roaming Activation, the NS1 SHALL send a ProfileAns message to the NS2 carrying Result=Success, RoamingActivationType.

The following describes the behavior when the RoamingActivationType is Passive.

Step 7:

If the Result of incoming ProfileAns indicates Success, or if the Steps 5 and 6 were skipped, then the NS2 SHALL send an PRStartReq message to the NS1 carrying the PHYPayload with Join-Request message ULMetadata.

Step 8:

When steps 5 and 6 are skipped, if there is no business agreement between the NS1 and the NS2, or if the NS1 could not identify the End-Device with the DevEUI, or if the End-Device is not allowed to perform Roaming Activation, or if the NS1 does not wish to enable Passive Roaming activation via NS2 then the NS1 shall proceed to step 10.

Otherwise, The NS1 SHALL send a JoinReq message to the JS carrying the PHYPayload with Join-request message, DevEUI, DevAddr, DLSettings, RxDelay, and optionally CFList defined by the NS1.

Step 9:

The JS processes the Join-request message and sends JoinAns to the NS1 carrying Result=Success, PHYPayload with Join-accept message, network session keys (SNwkSIntKey, FNwkSIntKey, and NwkSEncKey in case of a R1.1, and NwkSKey in case of a R1.0/1.0.2 End-Device), encrypted AppSKey or SessionKeyID or both, Lifetime in case of success, and Result=UnknownDevEUI or MICFailed in case of failure (e.g., if the End-Device is not recognized by the JS, or if the MIC of the Join-request fails the verification). Network session keys, and AppSKey are generated based on the LoRaWAN specification [LW10, LW102, LW11]. AppSKey is encrypted using a key shared between the JS and the AS when it is delivered from the JS to the NS.

Step 10:

If there is no business agreement between the NS1 and the NS2, then the NS1 SHALL send an PRStartAns message to the NS2 carrying Result=NoRoamingAgreement. If the NS1 could not identify the End-Device with the DevEUI, then the NS1 SHALL send a PRStartAns message to the NS2 carrying Result=UnknownDevEUI. If the End-Device is not allowed to perform Roaming Activation, then the NS1 SHALL send a PRStartAns message to the NS2 carrying Result= RoamingActDisallowed. If the NS1 does not wish to enable Passive Roaming activation via NS2, then it SHALL send a PRStartAns to the NS2 carrying Result=Deferred, and Lifetime. The NS2 SHALL not send any more PRStartReq to the NS1 for the same End-Device for the duration of Lifetime upon receiving this message.

Otherwise, the NS1 SHALL send a PRStartAns to the NS2 carrying the Result=Success, PHYPayload with Join-accept message , and Lifetime associated with the Passive Roaming. The NS1 SHALL also include DevEUI and ServiceProfile if NS2 is operating as a stateful fNS, and, FCntUp and FNwkSIntKey (in case of R1.1) or NwkSKey (in case of R1.0/1.0.2) in the PRStartAns message if NS1-NS2 Passive Roaming agreement requires the NS2 to perform MIC check on the uplink packets.

Step 11:

The NS2 SHALL forward the received PHYPayload with Join-accept message to the End-Device if PRStartAns message indicates success. The End-Device SHALL generate network session keys, and AppSKey based on the LoRaWAN specification [LW10, LW102, LW11] upon receiving the Join-accept message.

If encrypted AppSKey is not made available by the JS to the AS via the NS, then the AS SHALL retrieve it directly from the JS using the same method as defined in Step 8 of OTA Activation at Home Procedure (see Section 8).

When the procedure completes successfully, the NS2 becomes the fNS, and the NS1 becomes the sNS (in addition to being the hNS) of the newly created LoRa Session.

12.2.1 Packet Transmission

The details of uplink and downlink packet transmission between the sNS and the fNS after the two are engaged in Passive Roaming Activation for an End-Device are same as the Passive Roaming case as described in Section 11.3.2.

12.2.2 Passive Roaming Stop

Passive Roaming Stop Procedure (Section 11.3.3) is used when either the sNS or the fNS decides to terminate the roaming.

13 DevAddr Assignment

NetID is a 24bit network identifier assigned to LoRaWAN networks by the LoRa Alliance. Values 0x000000 and 0x000001 are reserved for experimental networks and networks that are not using roaming. These values can be used by any network without getting permission from the LoRa Alliance. LoRaWAN networks that use roaming need to obtain a unique NetID value assigned by the LoRa Alliance.

3 bits	21-N bits	N bits
Type	RFU	ID

Figure 16 NetID format

Figure 16 illustrates the format of the NetID which is composed of the following fields:

Type: The 3 MSB (Most Significant Bits) of the NetID indicates the NetID Type (0 through 7).

ID: Variable length LSB (Least Significant Bits) of NetID as assigned by the LoRa Alliance. Length of the ID field depends on the Type of the NetID.

RFU: If there are any unused bits in the NetID after the Type and ID fields are consumed, they are marked as RFU and set to zero. These RFU bits are placed in between the Type and ID bits, if those fields do not already consume the 24 bits of the NetID.

Table 2 provides the details on the Type field setting, number of RFU bits, and length of the ID field for each NetID Type.

NetID Type	24bit NetID		
	Type field setting (3 MSB)	Number of RFU bits	ID field
0	000	15	6 LSB
1	001	15	6 LSB
2	010	12	9 LSB
3	011	0	21LSB
4	100	0	21LSB
5	101	0	21LSB
6	110	0	21LSB
7	111	0	21LSB

Table 2 NetID Types

For example, the NetID value 0x000003 is a Type 0 NetID with ID=3, and value 0x6000FF is a Type 3 NetID with ID=255.

L bits	M bits	N bits
Type Prefix	NwkID	NwkAddr

Figure 17 DevAddr format

DevAddr is an End-Device identifier assigned by the LoRaWAN network. Figure 17 illustrates the format of the DevAddr which is composed of the following fields:

Type Prefix: Variable length MSB that indicates the NetID Type of the assigning network.

NwkID: Variable length bits that follow the Type Prefix field. They are used for identifying the network. The value of NwkID is set to the predefined number of LSB of ID field of the NetID.

NwkAddr: Variable length LSB that is assigned to the End-Device by the network.

Table 3 provides the details on the length and setting of Type Prefix field, size of NwkID and NwkAddr fields for each Type of NetID. The NS shall use the parameters defined in this table when assigning a DevAddr to its End-Devices based on its NetID.

NetID Type	32bit DevAddr			
	Type Prefix Length (MSB)	Type Prefix Value (binary)	Number of NwkID bits	Number of NwkAddr bits
0	1	0	6	25
1	2	10	6	24
2	3	110	9	20
3	4	1110	10	18
4	5	11110	11	16
5	6	111110	13	13
6	7	1111110	15	10
7	8	11111110	17	7

Table 3 DevAddr format based on the NetID Type

When number of NwkID bits is less than the number of bits in the ID field of the NetID (as in Types 3 through 7), that means multiple NetIDs are likely to map to the same NwkID value. Section 11.3 Passive Roaming describes how the fNS tries multiple NSs to find the sNS of the End-Device.

14 Periodic Recovery

Rejoin-request Type 1 message is defined for restoring connectivity with an End-Device in case of complete state loss on the sNS. The message is sent by the End-Device periodically for giving the sNS a chance to recover.

When an NS receives a Rejoin-request Type 1, the NS SHALL determine if it has a valid LoRa Session with the End-Device as identified by the received DevEUI. If the NS is not acting as the sNS for the End-Device, then the NS SHALL treat the incoming Rejoin-request Type 1 exactly same way as it would process a Join-request (i.e., following Activation at Home or Roaming Activation Procedures by transporting Rejoin-request message instead of the Join-request message from the NS to the JS). If the NS is acting as the sNS for the End-Device, then the NS SHALL behave as described in Section 6.2.4.4 of [LW11].

This procedure applies to only R1.1 [LW11] End-Devices and networks.

15 Rekeying and DevAddr Reassignment

If the sNS decides to either refresh the session keys, reset the frame counters, or assign a new DevAddr to the End-Device without changing the channel definitions, the sNS SHALL send a ForceRejoinReq with RejoinType 2 MAC command to the End-Device.

The End-Device SHALL send a Rejoin-request Type 2 message when it receives a ForceRejoinReq from the sNS.

The End-Device SHALL not send a Rejoin-request Type 2 message unless it receives a valid ForceRejoinReq with RejoinType 2 from its sNS. The sNS SHALL discard a received Rejoin-request Type 2 if the sNS has not sent a ForceRejoinReq with RejoinType 2 MAC command to the End-Device.

Processing of the Rejoin-request Type 2 message is same as processing of Rejoin-request Type 0 as described in Section 11.4.1 Handover Roaming Start, considering the receiving NS (NS2 in Figure 11) is already the sNS.

If the End-Device decides to refresh the session keys or reset the frame counters without receiving a ForceRejoinReq with RejoinType 2 MAC command from the sNS, then the End-Device SHALL send a Join-request.

This procedure applies to only R1.1 [LW11] End-Devices and networks.

16 Packet Metadata

16.1 UL Packet Metadata

Each uplink packet received by the LoRa system is associated with a set of parameters obtained from the radio receiver and the local context of the LoRa Session of the End-Device. Such parameters are shared among communicating network elements in the form of metadata along with the packet payload in order to assist uplink transmission. Table 4 illustrates the metadata details for the uplink packets.

Information element	Generated by	Carried over fNS-sNS interface	Carried over sNS-hNS interface	Description/notes
DevEUI	fNS	Yes	Yes	Included if available to the sender by means of the received packet or local context
DevAddr	fNS	Yes	Yes	Included if available to the sender by means of the received packet or local context
FPort	sNS	No	Yes	sNS sends FRMPayload (not PHYPayload) to the hNS, hence missing FPort is carried separately
FCntDown	sNS	No	Yes	The last downlink application counter used for the End-Device, if available. True 32 bits, if using 32-bit counters. Carries AFCntDown if using R1.1.
FCntUp	sNS	No	Yes	sNS sends FRMPayload (not PHYPayload) to the hNS, hence missing FCntUp is carried separately True 32 bits, if using 32-bit counters. Carries AFCntUp if using R1.1.
Confirmed	sNS	No	Yes	Set to True if MType is Confirmed Data Up, False otherwise
DataRate	fNS/sNS	Yes	Optional	Generated by the NS controlling the receiving GW
ULFreq	fNS/sNS	Yes	Optional	Transmission frequency of the UL packet. Generated by the NS controlling the receiving GW.
Margin	fNS/sNS	No	Optional	Reported by ReportDevStatus, if allowed by the Service Profile. Generated by the NS controlling the receiving GW.
Battery	fNS/sNS	No	Optional	Reported by ReportDevStatus, if allowed by the Service Profile. Generated by the NS controlling the receiving GW.
FNSULToken	fNS	Optional	No	Opaque value generated by the fNS, which encodes auxiliary parameters that can assist the fNS later with downlink packet transmission. (See Note 1)
RecvTime	fNS/sNS	Yes	Yes	Timestamp of the packet arrival (GPS time with 1sec precision). Generated by the NS controlling the receiving GW.
RFRegion	fNS	Yes	No	RFRegion of the fNS.
GWCnt	fNS/sNS	Optional	Optional	Number of Gateways that received the same UL packet within a pre-configured timeout period. Generated by the NS controlling the receiving GW.

GWInfo	fNS/sNS	Yes	Optional	List of parameters (see below) for each GW that received the same UL packet. Generated by the NS controlling the receiving GWs. Mandatory for fNS only if fNS can send DLs.
> ID	fNS	Optional	Optional	GW identifier
> RFRegion	fNS/sNS	Optional	Optional	RF region of the GW
> RSSI	fNS/sNS	Yes	Optional	Received signal strength indication
> SNR	fNS/sNS	Yes	Optional	Signal-to-noise ratio
> Lat	fNS/sNS	Optional	Optional	Latitude of the GW
> Lon	fNS/sNS	Optional	Optional	Longitude of the GW
> ULToken	fNS/sNS	Optional	No	Opaque value generated by the GW, which encodes auxiliary parameters that can assist the same GW later with downlink packet transmission. (See Note 1)
> DLAllowed	fNS/sNS	Yes	No	Indication from the GW about its resource availability for possible downlink transmission

Table 4 Uplink packet metadata

Note 1 : In case of stateless fNS, at least one of the two information elements SHALL be present.

16.2 DL Packet Metadata

Each downlink packet received or generated by the LoRa system is associated with a set of parameters obtained from the AS and the local context of the LoRa Session of the End-Device. Such parameters are shared among communicating network elements in the form of metadata along with the packet payload in order to assist the downlink transmission. Table 5 illustrates the metadata details for downlink packets.

Information element	Generated by	Carried over hNS-sNS interface	Carried over sNS-fNS interface	Description/notes
DevEUI	hNS	Yes	Optional	Not present in case of stateless fNS
FPort	hNS	Yes	No	hNS sends FRMPayload to sNS, hence FPort is carried separately. FPort=0 is disallowed. sNS SHALL return Result=InvalidFPort.
FCntDown	hNS	Yes	No	AFCntDown in R1.1
Confirmed	hNS/sNS	Yes	No	Optionally used for indicating Confirmed transmission
DLFreq1	sNS	No	Yes	Transmission frequency for RX1
DLFreq2	sNS	No	Yes	Transmission frequency for RX2
RXDelay1	sNS	No	Yes	Receive delay for RX1
ClassMode	sNS	No	Yes	Device mode for the DL
DataRate1	sNS	No	Yes	Data rate for RX1
DataRate2	sNS	No	Yes	Data rate for RX2
FNSULToken	sNS	No	Yes	Copy of the last FNSULToken received from the fNS, if available
GWInfo	sNS	No	Optional	List of ULToken parameters (see below) for each GW that received the latest UL packet. Values copied from the latest ULMetadata.
> ULToken	sNS	No	Yes	Copy of the ULToken received for each GW. If provided in ULMetadata, it SHALL be present in DLMetadata.
HiPriorityFlag	sNS	No	Yes	fNS SHOULD do its best to transmit the packet (e.g., set when sending RejoinSetupRequest command)

Table 5 Downlink packet metadata

17 Profiles

17.1 Device Profile

Device Profile includes End-Device capabilities and boot parameters that are needed by the NS for setting up the LoRaWAN radio access service. Table 6 illustrates the information elements that are included in a Device Profile. These information elements SHALL be provided by the End-Device manufacturer.

Information element	M/O	Description/notes
DeviceProfileID	M	ID of the Device Profile
SupportsClassB	M	End-Device supports Class B
ClassBTimeout	O	Maximum delay for the End-Device to answer a MAC request or a confirmed DL frame (mandatory if class B mode supported)
PingSlotPeriod	O	Mandatory if class B mode supported
PingSlotDR	O	Mandatory if class B mode supported
PingSlotFreq	O	Mandatory if class B mode supported
SupportsClassC	M	End-Device supports Class C
ClassCTimeout	O	Maximum delay for the End-Device to answer a MAC request or a confirmed DL frame (mandatory if class C mode supported)
MACVersion	M	Version of the LoRaWAN supported by the End-Device
RegParamsRevision	M	Revision of the Regional Parameters document supported by the End-Device
SupportsJoin	M	End-Device supports Join (OTAA) or not (ABP)
RXDelay1	O	Class A RX1 delay (mandatory for ABP)
RXDROffset1	O	RX1 data rate offset (mandatory for ABP)
RXDataRate2	O	RX2 data rate (mandatory for ABP)
RXFreq2	O	RX2 channel frequency (mandatory for ABP)
FactoryPresetFreqs	O	List of factory-preset frequencies (mandatory for ABP)
MaxEIRP	M	Maximum EIRP supported by the End-Device
MaxDutyCycle	O	Maximum duty cycle supported by the End-Device
RFRegion	M	RF region name
Supports32bitFCnt	O	End-Device uses 32bit FCnt (mandatory for LoRaWAN 1.0 End-Device)

Table 6 Device Profile

“M” in the M/O column indicates “Mandatory to include”, and “O” indicates “Optional to include”.

17.2 Service Profile

Service Profile includes service parameters that are needed by the NS for setting up the LoRa radio access service and interfacing with the AS. Table 7 illustrates the information elements that are included in a Service Profile.

Information element	Description/notes
ServiceProfileID	ID of the Service Profile
ULRate	Token bucket filling rate, including ACKs (packet/h)
ULBucketSize	Token bucket burst size
ULRatePolicy	Drop or mark when exceeding ULRate
DLRate	Token bucket filling rate, including ACKs (packet/h)
DLBucketSize	Token bucket burst size
DLRatePolicy	Drop or mark when exceeding DLRate
AddGWMetadata	GW metadata (RSSI, SNR, GW geoloc., etc.) are added to the packet sent to AS
DevStatusReqFreq	Frequency to initiate an End-Device status request (request/day)
ReportDevStatusBattery	Report End-Device battery level to AS
ReportDevStatusMargin	Report End-Device margin to AS
DRMin	Minimum allowed data rate. Used for ADR.
DRMax	Maximum allowed data rate. Used for ADR.
ChannelMask	Channel mask. sNS does not have to obey (i.e., informative).
PRAllowed	Passive Roaming allowed
HRAllowed	Handover Roaming allowed
RAAllowed	Roaming Activation allowed
NwkGeoLoc	Enable network geolocation service
TargetPER	Target Packet Error Rate
MinGWDiversity	Minimum number of receiving GWs (informative)

Table 7 Service Profile

17.3 Routing Profile

Routing Profile includes information that are needed by the NS for setting up data-plane with the AS. Table 8 illustrates the information elements that are included in a Routing Profile.

Information element	Description/notes
RoutingProfileID	ID of the Routing Profile
AS-ID	ID of the AS

Table 8 Routing Profile

18 Usage Data Records

18.1 Network Activation Record

Network Activation Record is used for keeping track of the End-Devices performing Roaming Activation. When the Roaming Activation Procedure takes place, then the NS SHALL generate a monthly Network Activation Record for each ServiceProfileID of another NS that has at least one End-Device active throughout the month, and dedicated Network Activation Records for each activation and deactivation of an End-Device from another NS. Table 9 illustrates the details of the Network Activation Record.

Information element	Description/notes
NetID	NetID of the roaming partner NS
ServiceProfileID	Service Profile ID
IndividualRecord	Indicates if this is an individual (de-)activation record (as opposed to cumulative record of End-Devices that are active throughout the month)
TotalActiveDevices	Number of End-Devices that have been active throughout the month. Included if this is a cumulative record.
DevEUI	DevEUI of the End-Device that has performed the (de-)activation. Included if this is an IndividualRecord for a (de-)activation event.
ActivationTime	Date/time of the activation. Included if this is an IndividualRecord for an activation event.
DeactivationTime	Date/time of the deactivation. Included if this is an IndividualRecord for a deactivation event.

Table 9 Network Activation Record

18.2 Network Traffic Record

Network Traffic Record is used for keeping track of the amount of traffic served for roaming End-Devices. The NS that allows roaming SHALL generate a monthly Network Traffic Record for each roaming type (Passive/Handover Roaming) under each ServiceProfileID of another NS that has at least one End-Device roaming into its network. Table 10 illustrates the details of the Network Traffic Record.

Information element	Description/notes
NetID	NetID of the roaming partner NS
ServiceProfileID	Service Profile ID
RoamingType	Passive Roaming or Handover Roaming
TotalULPackets	Number of uplink packets
TotalDLPackets	Number of downlink packets
TotalOutProfileULPackets	Number of uplink packets that exceeded ULRate but forwarded anyways per ULRatePolicy
TotalOutProfileDLPackets	Number of downlink packets that exceeded DLRate but forwarded anyways per DLRatePolicy
TotalULBytes	Total amount of uplink bytes
TotalDLBytes	Total amount of downlink bytes
TotalOutProfileULBytes	Total amount of uplink bytes that falls outside the Service Profile
TotalOutProfileDLBytes	Total amount of downlink bytes that falls outside the Service Profile

Table 10 Network Traffic Record

Packet and payload counters are only based on the user-generated traffic. Payload counters are based on the size of the FRMPayload field.

19 JoinEUI and NetID Resolution

A Network Server SHALL resolve the value of JoinEUI to the IP address and port number of the Join Server when it receives this value either in a Join-request or a Rejoin-request message. Similarly, NetID value SHALL be resolved to the IP address and port number of the associated Network Server when it is received by a Network Server in a Rejoin-request message.

Both types of address resolutions are carried out by using DNS. The solution mechanism is inspired by the "SIP: Locating SIP Servers", RFC 3263, and supports resolution of a single identifier to multiple alternative IP addresses and port numbers in order to support high availability and geo-redundancy.

It should be noted that some organizations need to operate Join Servers without operating a network, therefore the Join Server resolution mechanism needs to work without the need to allocate a NetID.

19.1 DNS configuration

The LoRa Alliance SHALL establish and operate two dedicated subdomains to resolve Join Servers and NetIDs, rooted at JOINEUIS.LORA-ALLIANCE.ORG and NETIDS.LORA-ALLIANCE.ORG, respectively.

A 24-bit NetID is represented as a name in the NETIDS.LORA-ALLIANCE.ORG domain by a sequence of nibbles with the suffix ".NETIDS.LORA-ALLIANCE.ORG". The high-order nibble is encoded first, followed by the next higher-order nibble and so on. Each nibble is represented by a hexadecimal digit. For example, the domain name corresponding to the NetID

    1290 (0x00050A)

would be

    00050a.NETIDS.LORA-ALLIANCE.ORG

A Join EUI (IEEE EUI-64) is represented as a name in the JOINEUIS.LORA-ALLIANCE.ORG domain by a sequence of nibbles separated by dots with the suffix ".JOINEUIS.LORA-ALLIANCE.ORG". The sequence of nibbles is encoded in reverse order, i.e., the low-order nibble is encoded first, followed by the next low-order nibble and so on. Each nibble is represented by a hexadecimal digit. For example, the domain name corresponding to the EUI

    00-00-5E-10-00-00-00-2F

would be

    f.2.0.0.0.0.0.0.0.1.e.5.0.0.0.0.JOINEUIS.LORA-ALLIANCE.ORG

The NAPTRs SHALL point to replacement servers according to order, preference, and flags, and service parameters provided by the operators.

For example:

	order	pref	flags	service	regexp	replacement
IN NAPTR	50	50	"S"	"LWN"	""	_LWN._X2H.operator.com.
IN NAPTR	90	50	"S"	"LWNS"	""	_LWN._J2H.operator.com
IN NAPTR	100	50	"S"	"LWN"	""	_LWN._B2H.operator.com.

Lower values of “order” field have higher precedence. For same order values, lower values of “pref” field have higher precedence. Same priority records in the lexicographic order of <order>/<pref> SHALL be load balanced.

The NAPTR record “service” describes the service provided by the server, as well as the transport protocol that may be used to access it. “LWN” denotes a LoRaWAN Server (either Join Server or a Network Server) using HTTP, and “LWNS” denotes a server using HTTPS.

There are currently four possible flags. "S" denotes that an SRV lookup is to be performed on the output of this NAPTR record. "A" means the result should be looked up as an "A", "AAAA" or "A6" record. A "U" means that the NAPTR result is an absolute URI that the application should process. A "P" would signify a "non-terminal" rule where additional NAPTR lookups would be necessary.

Each NAPTR records can be used to transform the “domain” value for the next resolution step.

When flag “S” is used, the next resolution step will lookup an SRV record, as defined in RFC 2782, as in the example below. Operator SHALL include corresponding SRV records in its own DNS.

		Priority	Weight	Port	Target
IN	SRV	0	1	443	server1.operator.com
IN	SRV	0	2	443	server2.operator.com

It is not necessary for the domain suffixes in the NAPTR replacement field to match the domain of the original query (i.e., operator.com above).

19.2 NetID Resolution

The input parameter is the 24-bit NetID as carried in the Rejoin-request message sent by the End-Device to the Network Server of the Visited Network.

The Visited Network Server SHALL perform a DNS query for NAPTR records using the mapping for NetIDs described in Section 19.1. The Network Server performing the query SHALL eliminate results with transport and encoding protocols that are not supported by the server itself.

Network Server SHALL perform DNS recursively until the IP address and port number of the Home Network Server is resolved.

19.3 JoinEUI Resolution

The input parameter is the 64-bit JoinEUI as carried in the Join-request message sent by the End-Device to the Network Server of the Home Network or the Rejoin-request message sent by the End-Device to the Network Server of the Visited Network.

The receiving Network Server SHALL perform a DNS query for NAPTR records using the mapping for JoinEUIs described in Section 19.1. The Network Server performing the query SHALL eliminate results with transport and encoding protocols that are not supported by the server itself.

Network Server SHALL perform DNS recursively until the IP address and the port number of the Join Server is resolved.

20 Transport Layer

The LoRaWAN backend interfaces involve interconnection among the network elements, such as the JS, the NS, and the AS for delivering control signals and data packets. The following network interfaces are in scope of the current specification:

- AS-JS (optional)
- JS-NS
- NS-NS

A JoinEUI identifies a JS, whereas an NS is identified by its NetID. Multiple JoinEUIs may identify the same JS. Both the JoinEUI and the NetID can be resolved into the IP address and port number of the respective servers by using DNS.

Network elements SHALL rely on a security solution that can provide mutual end-point authentication, integrity and replay protection, and confidentiality when communicating with each other. The choice of mechanism used for achieving these properties is left to the deployments (e.g., using IPsec VPN, HTTPS, physical security, etc.)

Network element SHALL use HTTP 1.1 [RFC2616] and encode the payloads using JSON. In order to support sending messages (signal or data) in both directions, a pair of HTTP connections needs to be setup between the two end-points. Each end-point SHALL initiate and maintain an HTTP connection with the other end-point. HTTP end-points SHOULD use persistent connections.

21 Key Transport Security

Several times during a LoRa Session, keys need to be exchanged between servers (on JS-AS, JS-NS or NS-NS interfaces for instance).

To secure the transport of those keys, Key Encryption Keys (KEK) can be used to encrypt them, following the wrapping process defined in the RFC 3394.

On top of that, each Key Encryption Key is associated with a Key Encryption Key Label (KEKLabel) and a wrapping algorithm as defined in the RFC3394 to allow selecting the right key and the right algorithm during the unwrapping operation.

The set of KEK, associated KEKLabels, and algorithm are generated and exchanged between the servers during an offline process that is not part of this specification, servers being of 2 kind: the key requester and the key sender.

The decision to wrap or not a key SHALL always be taken by the entity who is in charge of delivering the key (i.e., key sender).

Table 11 provides the details of the KeyEnvelope Object that is used for wrapping keys.

Information element	M/O	Description/notes
KEKLabel	O	This label identifies the key to be used to unwrap the AESKey. If this value is not present, it means the AESKey is transmitted in clear.
AESKey	M	AESKey carries the RFC3394-wrapped key if the KEKLabel field is present. If the KEKLabel field is not present, then the AESKey carries the key in clear.

Table 11 KeyEnvelope Object

22 Messages and Payloads

22.1 Encoding

HTTP is used as the transport layer for sending the backend request and answer messages (e.g., JoinReq and JoinAns). Following interfaces carry both the backend request and answer messages over HTTP Requests while using HTTP Responses simply for acknowledging the delivery: fNS-sNS, sNS-hNS. Following interfaces carry the backend request messages over HTTP Requests, whereas the backend answer messages may be carried over either the HTTP Response or a subsequent HTTP Request: hNS-JS, vNS-JS, AS-JS. The method used by the JS for each backend peer is determined out-of-band.

Network elements SHALL use JSON data format for sending request and answer messages. When a network element has a message to send to another network element in HTTP Request, it SHALL generate an HTTP POST Request for Target URL. Target URL is a configuration parameter that is agreed upon between the two network elements interfacing with each other. For example, on a given NS, the Target URL for a JS can be "https://js.lora_operator.com".

HTTP carries the request and answer messages as a JSON-encoded payload with various objects. Names of the objects that need to be included in a given request or answer message are provided in the sections that describe the detailed message flows. Encoding of each object type is provided in Section 22.3. Each message SHALL include a ProtocolVersion object whose value is set to “1.0” by the implementations of this specification, MessageType object that defines the action required for that message, and SenderID and ReceiverID objects. The sender of the message SHALL set the SenderID to the NetID, JoinEUI, or AS-ID of the sender, depending on whether the sender is an NS or JS or an AS, respectively. Similarly, the sender of the message SHALL set the ReceiverID to the NetID, JoinEUI, or AS-ID of the intended receiver, depending on whether the receiver is an NS or JS or an AS, respectively.

In order for a network element to be able to match a received answer message with the pending request message a TransactionID is used. The sender of a request message SHALL include a TransactionID in the message whose value setting is at the discretion of the sender. The sender of an answer message SHALL include the same TransactionID that was recevied in the request message that triggered the answer message. If a network element receives an answer message for which there is no pending request with the TransactionID value, then it SHALL discard the received message.

If the ProtocolVersion of the received message is not set to “1.0”, then the receiving network element SHALL return a message carrying Result=InvalidProtocolVersion. If the SenderID or the ReceiverID of the received message is unknown to the receiving network element, then it SHALL return a message carrying Result=UnknownSender or UnknownReceiver.

A network element MAY include SenderToken in its messages if it expects the target network element to echo the same value in ReceiverToken for each subsequent messages that are associated with the same End-Device. The sNS SHALL NOT send a SenderToken when communicating with a stateless fNS, as the fNS cannot store that token. A network element SHALL include a ReceiverToken in its messages if it received a SenderToken from the target network element for the same End-Device. In that case the network element SHALL copy the value of the received SenderToken to the transmitted ReceiverToken.

Figure 18 and Figure 19 illustrate two variants of the HTTP message flow for OTA Activation at Home Procedure as an example. While these figures are showing the HTTP details, rest of the figures in this document only illustrate the backend messages (e.g., not showing HTTP Responses unless they carry a backend message as a payload).

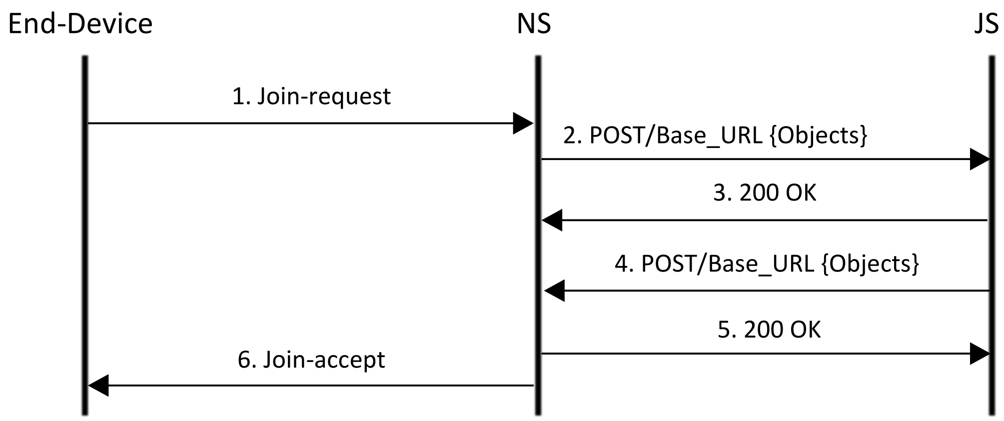

Figure 18 Backend messages carried over HTTP Requests

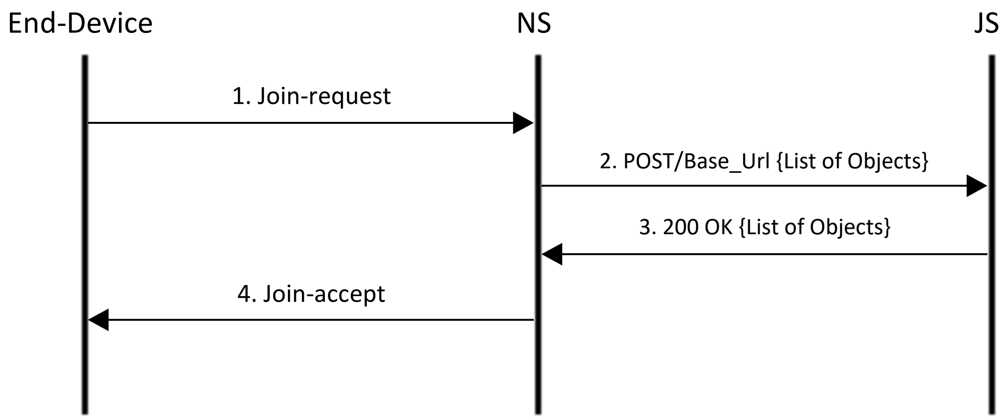

Figure 19 Backend messages carried over HTTP Request and Responses

22.2 Backend Message Types

Table 12 provides the list of backend message types in pairs.

Message Types	
JoinReq	JoinAns
RejoinReq,	RejoinAns
AppSKeyReq	AppSKeyAns
PRStartReq	PRStartAns
PRStopReq	PRStopAns
HRStartReq	HRStartAns
HRStopReq	HRStopAns
HomeNSReq	HomeNSAns
ProfileReq	ProfileAns
XmitDataReq	XmitDataAns

Table 12 Backend message types

Table 13 provides the list of payload objects carried by each backend message. If a discrepancy ever occurs between the Table 13 and the description of the associated procedures, the latter one takes precedence.

	JoinReq	JoinAns	RejoinReq	RejoinAns	AppSKeyReq	AppSKeyAns	PRStartReq	PRStartAns	PRStopReq	PRStopAns	HRStartReq	HRStartAns	HRStopReq	HRStopAns	HomeNSReq	HomeNSAns	ProfileReq	ProfileAns	XmitDataReq	XmitDataAns
ProtocolVersion	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
SenderID	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
ReceiverID	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
TransactionID	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
MessageType	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
SenderToken	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
ReceiverToken	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
MACVersion	M		M								M									
PHYPayload	M	Ms	M	Ms			M				M	Ms							M[1]	
FRMPayload																			M[1]	
Result		M		M		M		M		M		M		M		M		M		M
DevEUI	M		M		M	M		Os	M				M		M		M			
Lifetime		Ms		Ms				Ms	O			Ms								
SNwkSIntKey		Ms[1a]		Ms[1a]								Ms[1a]								
FNwkSIntKey		Ms[1a]		Ms[1a]				Os[1]				Ms[1a]								
NwkSEncKey		Ms[1a]		Ms[1a]								Ms[1a]								
NwkSKey		Ms[1b]		Ms[1b]				Os[1]				Ms[1b]								
FCntUp								Os												
DevAddr	M		M								M									
DeviceProfile											M	Of						Ms		
ServiceProfile								Os				Ms								
ULMetaData							M				M								M[2]	
DLMetaData												Ms							M[2]	
DLSettings	M		M								M									
RxDelay	M		M								M									
CFList	O		O								O									
AppSKey		Ms[1]		Ms[1]		Ms														
SessionKeyID		Ms[1]		Ms[1]	M	M														
DeviceProfileTimestamp											M	Of						Ms		
HNetID																Ms				
FCntDown																				
RoamingActivationType																		Ms		
DLFreq1																				Os
DLFreq2																				Os
VSExtension	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O

Table 13 Messages and payloads

The following notations are used in Table 13:

M: Mandatory
O : Optional
Ms : Mandatory, when Result=Success
Mf : Mandatory, when Result=Failure
Os : Optional, when Result=Success
Of : Optional, when Result=Failure
M^X : Mandatory to include exactly one of the 2 (groups of) objects marked with the same value X. When shown as M^{XY}, objects marked with the same value Y are considered as a group.
An empty cell indicates the object is never used with the designated message.

22.3 Data Types

Table 14 provides the JSON object details for various message payloads defined in this specification. When an object defined in this specification corresponds to a parameter defined in the LoRaWAN specification (e.g., DevEUI, SNwkSIntKey, etc.), then the parameter details in that specification also apply to the corresponding object value in this specification (e.g., DevEUI is 64 bits, SNwkSIntKey is 128 bits, etc.).

The object named VSExtension (Vendor-Specific Extension) allows carrying proprietary objects between the servers as needed in specific deployment scenarios. Definition of its content is left to specific implementations. The server SHALL process a received VSExtension Object if it is recognized by the server, and discard it otherwise.

Object Name	Value Type	Notes
ProtocolVersion	String	Version of backend specification. E.g., "1.0".
SenderID	String	Hexadecimal representation in ASCII format in case of carrying NetID or JoinEUI, ASCII string in case of AS-ID
ReceiverID	String	Hexadecimal representation in ASCII format in case of carrying NetID or JoinEUI, ASCII string in case of AS-ID
TransactionID	Number	32bit value
MessageType	String	String representation of values in Table 12 (e.g., “JoinReq”)
SenderToken	String	Hexadecimal representation in ASCII format
ReceiverToken	String	Hexadecimal representation in ASCII format
PHYPayload	String	Hexadecimal representation in ASCII format
FRMPayload	String	Hexadecimal representation in ASCII format
Result	Object	See Table 15
DevEUI	String	Hexadecimal representation in ASCII format
Lifetime	Number	Unit: Seconds
SNwkSIntKey	Object	See Table 16
FNwkSIntKey	Object	See Table 16
NwkSEncKey	Object	See Table 16
NwkSKey	Object	See Table 16
DevAddr	String	Hexadecimal representation in ASCII format
HNetID	String	Hexadecimal representation in ASCII format
DeviceProfile	Object	See Table 17

ServiceProfile	Object	See Table 18
RoutingProfile	Object	See Table 19
ULMetaData	Object	See Table 20
DLMetaData	Object	See Table 22
DLSettings	String	Hexadecimal representation in ASCII format
RxDelay	Number	
CFList	String	Hexadecimal representation in ASCII format
AppSKey	Object	See Table 16
SessionKeyID	String	Hexadecimal representation in ASCII format
DeviceProfileTimestamp	String	Timestamp of last Device Profile change (ISO 8601)
RoamingActivationType	String	Acceptable values: “Passive”, “Handover”
VSExtension	Object	See Table 23

Table 14 JSON encoding of top-level objects

Hexadecimal ASCII printable representation of a value may start with "0x" and may use upper or lower case letters.

Table 15 provides the details of the Result Object.

Object Name	Value Type	Notes
ResultCode	String	"Success" or one of the error strings defined in Table 24
Description	String	Detailed information related to the ResultCode (optional).

Table 15 Result Object

Table 16 provides the details of the KeyEnvelope Object. This object format is used for SNwkSIntKey, FNwkSIntKey, NwkSEncKey, NwkSKey, and AppSKey Objects.

Object Name	Value Type	Notes
KEKLabel	String	
AESKey	String	Hexadecimal representation in ASCII format

Table 16 KeyEnvelope Object

Table 17 provides the details of the DeviceProfile Object.

Object Name	Value Type	Notes
DeviceProfileID	String	
SupportsClassB	Boolean	
ClassBTimeout	Number	Unit: seconds
PingSlotPeriod	Number	
PingSlotDR	Number	
PingSlotFreq	Number	
SupportsClassC	Boolean	
ClassCTimeout	Number	Unit: seconds
MACVersion	String	Example: "1.0.2" [LW102]
RegParamsRevision	String	Example : "B" [RP102B]
RXDelay1	Number	
RXDROffset1	Number	
RXDataRate2	Number	Unit: bits-per-second
RXFreq2	Number	Value of the frequency, e.g., 868.10
FactoryPresetFreqs	Array of Numbers	
MaxEIRP	Number	In dBm
MaxDutyCycle	Number	Example: 0.10 indicates 10%
SupportsJoin	Boolean	
RFRegion	String	See Note 2
Supports32bitFCnt	Boolean	

Table 17 DeviceProfile Object

Note 2: Valid string values include “EU868”, “US902”, “China779”, “EU433”, “Australia915”, “China470”, “AS923”.

Table 18 provides the details of the ServiceProfile Object.

Object Name	Value Type	Notes
ServiceProfileID	String	
ULRate	Number	
ULBucketSize	Number	
ULRatePolicy	String	Acceptable values: "Drop", "Mark"
DLRate	Number	
DLBucketSize	Number	
DLRatePolicy	String	Acceptable values: "Drop", "Mark"
AddGWMetadata	Boolean	
DevStatusReqFreq	Number	Unit: requests-per-day
ReportDevStatusBatery	Boolean	
ReportDevStatusMargin	Boolean	
DRMin	Number	
DRMax	Number	
ChannelMask	String	Hexadecimal representation in ASCII format
PRAllowed	Boolean	
HRAllowed	Boolean	
RAAllowed	Boolean	
NwkGeoLoc	Boolean	
TargetPER	Number	Example: 0.10 indicates 10%
MinGWDiversity	Number	

Table 18 ServiceProfile Object

Table 19 provides the details of the RoutingProfile Object.

Object Name	Value Type	Notes
RoutingProfileID	String	
AS-ID	String	Value can be IP address, DNS name, etc.

Table 19 RoutingProfile Object

Table 20 provides the details of the ULMetaData Object.

Object Name	Value Type	Notes
DevEUI	String	Hexadecimal representation in ASCII format, big-endian, no separator
DevAddr	String	Hexadecimal representation in ASCII format
FPort	Number	Integer
FCntDown	Number	Integer
FCntUp	Number	Integer
Confirmed	Boolean	
DataRate	Number	See data rate tables in Regional Parameters document
ULFreq	Number	Floating point (MHz)
Margin	Number	Integer value reported by the End-device in DevStatusAns
Battery	Number	Integer value reported by the End-device in DevStatusAns
FNSULToken	String	Hexadecimal representation in ASCII format
RecvTime	String	Use ISO 8601
RFRegion	String	See Note 2 (above)
GWCnt	Number	Integer
GWInfo	Array of GWInfoElement Objects	See Table 21

Table 20 ULMetadata Object

Table 21 provides the details of the GWInfoElement Object.

Object Name	Value Type	Notes
ID	String	Hexadecimal representation of 32bit value in ASCII
RFRegion	String	See Note 2 (above)
RSSI	Number	Signed integer, unit: dBm
SNR	Number	Unit: dB
Lat	Number	
Lon	Number	
ULToken	String	Hexadecimal representation in ASCII format
DLAllowed	Boolean	

Table 21 GWInfoElement Object

Table 22 provides the details of the DLMetaData Object.

Object Name	Value Type	Notes
DevEUI	String	Hexadecimal representation in ASCII format
FPort	Number	
FCntDown	Number	
Confirmed	Boolean	
DLFreq1	Number	At least DLFreq1 or DLFreq2 SHALL be present.
DLFreq2	Number	At least DLFreq1 or DLFreq2 SHALL be present.
RXDelay1	Number	
ClassMode	String	Only values "A" and "C" are supported
DataRate1	Number	Present only if DLFreq1 is present
DataRate2	Number	Present only if DLFreq2 is present
FNSULToken	String	Hexadecimal representation in ASCII format
GWInfo	Array of GWInfoElement Objects	See Table 21
HiPriorityFlag	Boolean	

Table 22 DLMetadata Object

Table 23 provides the details of VSExtension Object.

Object Name	Value Type	Notes
VendorID	String	OUI of the vendor, hexadecimal representation in ASCII format
Object	opaque	The nature of the object is not defined

Table 23 VSExtension Object

22.4 Result Codes

Table 24 provides list of values that can be assigned to the Result Object.

Value	Description
"Success"	Success, i.e., request was granted
"MICFailed"	MIC verification has failed
"JoinReqFailed"	JS processing of the JoinReq has failed
"NoRoamingAgreement"	There is no roaming agreement between the operators
"DevRoamingDisallowed"	End-Device is not allowed to roam
"RoamingActDisallowed"	End-Device is not allowed to perform activation while roaming
"ActivationDisallowed"	End-Device is not allowed to perform activation
"UnknownDevEUI"	End-Device with a matching DevEUI is not found
"UnknownDevAddr"	End-Device with a matching DevAddr is not found
"UnknownSender"	SenderID is unknown
"UnkownReceiver"	ReceiverID is unknown
"Deferred"	Passive Roaming is not allowed for a period of time
"XmitFailed"	fNS failed to transmit DL packet
"InvalidFPort"	Invalid FPort for DL (e.g., FPort=0)
"InvalidProtocolVersion"	ProtocolVersion is not supported
"StaleDeviceProfile"	Device Profile is stale
"MalformedRequest"	JSON parsing failed (missing object or incorrect content)
"FrameSizeError"	Wrong size of PHYPayload or FRMPayload
"Other"	Used for encoding error cases that are not standardized yet

Table 24 Valid values for Result Object

When used, Description field of Result Object optionally reveals the error details.

Glossary

ABP	Activation by Personalization
ADR	Adaptive Data Rate
API	Application Programming Interface
AS	Application Server
DNS	Domain Name Server
ED	End-device
fNS	Forwarding Network Server
GW	LoRa Gateway
HTTP	HyperText Transfer Protocol
hNS	Home Network Server
IP	Internet Protocol
JS	Join Server
JSON	JavaScript Object Notation
KEK	Key Encryption Key
LoRa™	Long Range modulation technique
LoRaWAN™	Long Range network protocol
MAC	Medium Access Control
MIC	Message Integrity Code
NAPTR	Naming Authority Pointer
NS	Network Server
OTA	Over-the-Air
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
SF	Spreading Factor
SIP	Session Initiation Protocol
SNR	Signal-to-Noise Ratio
sNS	Serving Network Server

Bibliography

References

[LW10] LoRaWAN Specification, Version 1.0, LoRa Alliance, January 2015.
[LW102] LoRaWAN Specification, Version 1.0.2, LoRa Alliance, July 2016.
[RP102B] LoRaWAN 1.0.2 Regional Parameters, Revision B, LoRa Alliance, Feb 2017.
[LW11] LoRaWAN Specification, Version 1.1, LoRa Alliance, October 2017.

NOTICE OF USE AND DISCLOSURE

LoRa Alliance, Inc.
3855 SW 153rd Drive
Beaverton, OR 97007

Note: All Company, brand and product names may be trademarks that are the sole property of their respective owners.