How to Use LoRa Basics™ Station

Semtech

September, 2020

LoRa Basics™ Station is a LoRaWAN® gateway software project initially released by Semtech in 2019. In this article we answer a few questions you may ask about this project:

- Why LoRa Basics Station?
- What Can LoRa Basics Station Do?
- How is LoRa Basics Station Built?
- How Does LoRa Basics Station Work?
- How Secure is LoRa Basics Station?
- How Can I Get Started?

Why LoRa Basics Station?

In LoRaWAN networking, the gateway is the physical layer (PHY) interface for the LoRaWAN Network Server (LNS): It listens to certain parts of the radio spectrum, decodes valid LoRaWAN packets out of signals originating from sensors using LoRa modulation and forwards them to the LNS. It also transmits LoRaWAN packets from the LNS down to a sensor as LoRa-modulated signals. Read this blog post in the LoRa Developer Portal's Tech Journal to learn more about the role of the gateway in LoRaWAN networks.

LoRa Basics Station (Station) is a LoRaWAN gateway software implementation which provides this core functionality in terms of handling the packet flow, managing spectrum access and LNS backhaul connectivity, and more. To accomplish these tasks securely, reliably and efficiently over a large gateway population, Station defines two back-end protocols:

1) The **LNS protocol** is the primary data plane, providing a low-latency bidirectional communication channel over secure WebSockets. Aspects of load-balancing and centralized configuration management are built into this protocol.

2) In addition, Station provides credentials management and a firmware update interface via the **Configuration and Update Service (CUPS) protocol** – a simple authenticated HTTPS transaction for delivering LNS interface credentials and signed firmware binaries.

Documentation and Protocol Definitions: https://lora-developers.semtech.com/resources/tools/lora-basics/lora-basics-for-gateways/

Gateway Software Implementation: https://github.com/lorabasics/basicstation

What Can LoRa Basics Station Do?

LoRa Basics Station fulfils all tasks related to basic packet forwarding functionality for LoRaWAN Class A, B and C. In addition, Station has several features that make it particularly advantageous for large scale gateway deployments where centralized configuration management and remote inspection capabilities are key. Station's architecture makes it easy to port to different platforms, or even embedded systems. The CUPS and LNS protocol's base capabilities are extensible and facilitate orchestration of more complex usage scenarios around configuration management, time keeping, remote inspection and intervention, and more.

Feature overview:

- Support for common radio hardware reference designs:
    - v1.5 (single sx1301 over SPI + FPGA for LBT)
    - v2 (multiple sx1301s over SPI + DSP for fine time stamping)
    - Picocell (like v1.5 but over USB and reduced power consumption – USB dongle)
    - Corecell (single sx1302 over SPI)
- Lean architecture:
    - Resource efficient design - ready for embedded applications
    - Minimal third-party dependencies for best portability
    - Low-level fine tuning possible via rich set of runtime parameters
- Full support for Linux hosts:
    - Daemon mode
    - Flexible logging, log file rotation and truncation
    - Master/slave setup for multi-radio operation
- Secure and firewall-friendly TCP/IP communication:
    - No ingress connections needed
    - Authentication via TLS certificates or HTTP token headers
- LNS Protocol:
    - Centralized radio parameter management
    - Remote system commands and optional interactive shell
    - Flexible health and status reporting mechanism
    - GPS time inference
    - Time transfer (facilitating indoor Class B use cases)
- CUPS Protocol:
    - Transactional update of connection credentials with roll-back capability
    - Secure firmware update delivery with ECDSA signatures

For a more comprehensive description of features, see the documentation on the LoRa Developer Portal.

How is LoRa Basics Station Built?

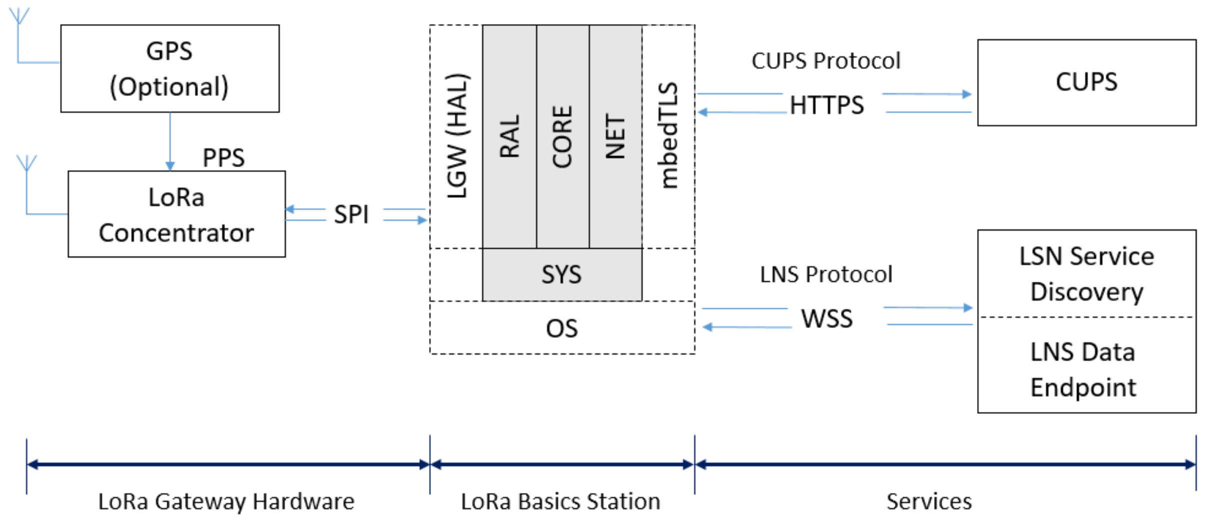

Figure 1 LoRa Basics Station System Overview
Source: ***https://lora-developers.semtech.com/resources/tools/lora-basics/lora-basics-for-gateways/***

The compiled Station binary is an application which is executed in the user space of the operating system of the gateway's host platform. Figure 1 illustrates how Station interacts with the other system components.

Towards the left, the diagram hints at the gateway radio hardware and its interfaces to the gateway's host platform, which is typically connected via a serial interface like SPI. Station builds on top of the hardware abstraction layer (HAL) libraries that Semtech provides for different gateway radio hardware reference designs to interact with radio hardware.

Towards the right of the image, the back-end services are shown which are expected by Station together with their corresponding protocols LNS and CUPS. The connection to these services is typically established via an IP-based network. This allows the CUPS and LNS protocols to build on top of a proven protocol stack used all over the world wide web, namely HTTP and WebSockets for transport and TLS for security.

The center block represents the gateway's host platform running the Station process inside the operating system (OS). The block is split into the high-level modular components of Station, which provide distinct interfaces to the various system components:

- RAL: Radio abstraction layer for different HALs
- SYS: System abstraction layer for different operating systems (Linux, FreeRTOS, etc.)
- NET: Networking abstraction layer built on top of mbedTLS

Figure 2 shows a more detailed view of the LoRa Basics Station software architecture.

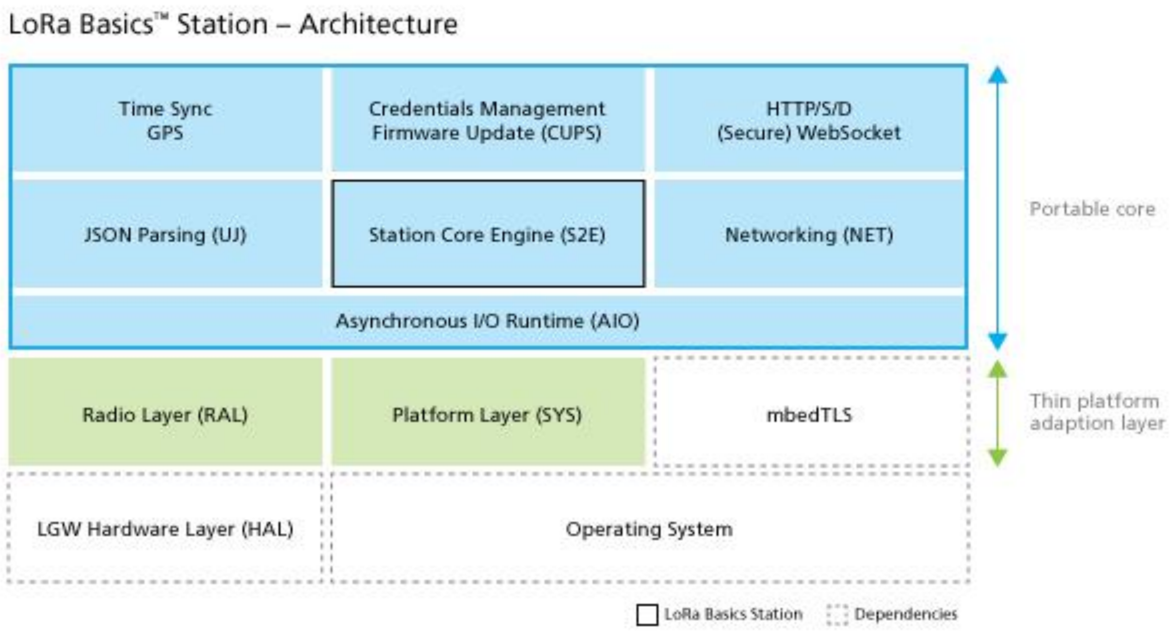

Figure 2: LoRa Basics Station Architecture

The block diagram reveals that the RAL, SYS and mbedTLS modules comprise a hardware-specific adaptation layer which provides a unified interface for the system components towards the portable core implementation. At compile-time, the choice for the RAL and SYS layers is made and the resulting object is linked statically with the mbedTLS library.

The portable core is a dependency-free C module built around an asynchronous cooperative multitasking runtime (AIO) and comprises all the core functionality of Station: LoRa packet handling, packet buffering, downlink queue management, spectrum access management, protocol parsing (JSON), protocol state logic (CUPS/LNS), time synchronization, and more.

This architecture enables:

- Ease of portability: The effort of porting Station to a new HAL is confined to changes to the RAL module. The effort of porting Station to a new host platform is confined to changes to the SYS module.
- Ease of testability: Hardware-independent testing can be done by replacing the HAL block with a component which translates the HAL API calls of the RAL with remote procedure calls (RPCs) into the test harness. The LoRa Basics Station regression tests make use of this approach.

The Station build environment allows us to set compile-time options which yield an optimal Station executable for the target environment. These options are grouped into the high-level identifiers *platform* and *variant* as defined in **setup.gmk**. Dependencies are automatically pulled, compiled and linked during the build process.

How Does LoRa Basics Station Work?

Configuring LoRa Basics Station:

In order to execute the Station process, a few configuration files have to be set up. The **station.conf** file contains settings to configure aspects of station as well as static configuration of the radio hardware:

```
{
   "radio_conf": { .. }
   "station_conf": { .. }
}
```

The **radio_conf** object depends on the radio hardware configuration. It is populated once by the manufacturer and does not change over time. The station_conf object defines a number of behavioral parameters. Refer to LoRa Basics Station: Configuration Files for all the details on configuring Station.

Before starting the Station process, we need to define where we need to connect. Here we have two options:

1) The file **tc.uri** contains the full URL to the LNS WebSocket endpoint. The URL starts with **ws://** in case a plain text connection is used. Using the **wss://** scheme will trigger a TLS connection based on the **tc.{cert,key,trust}** credentials set.
2) If the file **cups.uri** is present, Station will automatically use the CUPS protocol to retrieve the LNS connection details. This file contains a valid URL to the CUPS HTTP endpoint. Using the **https://** scheme will trigger a TLS connection based on the **cups.{cert,key,trust}** credentials set.

After a successful connection is established to any of these endpoints, the corresponding files are copied to {tc,cups}-bak.{uri,cert,key,trust} files and used as fall-back connections during a CUPS transaction. If the optional **{tc,cups}-boot.{uri,cert,key,trust}** files are present in Station's home directory, these will be used as ultimate fall-back endpoints in case the others fail.

The LNS Protocol

Assuming proper configuration of the LNS connection via **tc.uri**, a Station will connect to it and, in a first step, advertise itself to the service endpoint. The service endpoint responds with a final WebSocket connection endpoint, where Station connects right away using the same credentials. After advertising itself to the final connection endpoint, Station receives a configuration object from the LNS which contains, at minimum, the frequency plan and regional context in which the gateway operates.

After the frequency plan has been applied and the radio is started, the gateway is in a steady state, during which time it can exchange a number of messages. Whenever a valid LoRaWAN frame is demodulated by the radio, it is forwarded directly to the LNS. Additionally, the LNS can transmit downlinks via the gateway to an end device. If the downlink is sent, an acknowledgement is returned to the LNS. If the LNS connection terminates unexpectedly, any received uplinks are buffered until the connection is re-established.

The CUPS Protocol

The CUPS protocol is a separate engine within Station. It uses HTTP POST with a JSON-encoded body to declare its current configuration state to the CUPS endpoint. The CUPS server uses this information to decide whether an update is necessary and responds with and octet-stream containing a new set of credentials for the CUPS endpoint or the LNS endpoint or both. An empty response means that no update is necessary.

Within the same response, the CUPS may choose to append a signed executable blob. The format of this executable is not relevant to the protocol itself but it should match a format which the gateway host platform is able to apply. It could be a full firmware image in the case of an embedded host platform or an executable script, or an auto-extraction archive in the case of a Linux-based host system. For checking the signature of the executable, multiple public ECDSA signing keys can be configured by placing them into sig-{0,1,2,…}.key files.

CUPS is not designed to be a comprehensive gateway manager. Instead, its underlying primitives offer a simple unified approach to the most basic management tasks: credential management and secure remote firmware updates.

How Secure is LoRa Basics Station?

For securing the IP-based back-end connections, LoRa Basics Station relies on well-known concepts used daily in the World Wide Web. The authentication mode for a given connection depends on the presence and content of the connection definition files {tc,cups}.{uri,trust,key,crt}.

No Authentication

No authentication is used when the service endpoint URLs specifically declare plain text connections:

***.uri**: `ws://...` (for tc.uri) or `http://...` (for cups.uri)

This is useful when the security is already implemented by the operating system on the network layer by means of IPsec or if security is established by firewalls and network separation where all hosts are trusted. Gateway connections over the internet without IPsec should always use one of the next authentication methods.

Server and Client Authentication

Mutual authentication is enabled in case the service endpoint URL is declared as a secure connection:

***.uri**: `wss://...` (for tc.uri) or `https://...` (for cups.uri)

The server's authenticity is checked against a certificate authority (CA). The certificate of the CA must be provided in the *.trust file.

> ***.trust**: Trusted certificate of the server's CA (PEM or DER encoded X509 certificate)

Stations provide two options for establishing client authenticity depending, on the presence of *.key and *.crt files:

1. Client token authentication [*.key]

   ***.key**: Valid HTTP headers containing client authentication information for the server (e.g. `Authorization: ...\r\n`)

2. Client certificate authentication [*.crt, *.key]

   ***.crt**: Client certificate (PEM or DER encoded X509 certificate)

   ***.key**: Private key (PEM or DER encoded x509 key)

Hint: A good way to debug the TLS setup of your gateway is to execute Station with the environment variable `STATION_TLSDBG` set to a verbosity level between 1 (lowest) and 4 (highest).

How Can I Get Started?

Getting started with LoRa Basics Station is easy. You can jump right in by going directly to the Quick Start Guide. If you want to get into the details, there is a wealth of information about LoRa Basics Station on the LoRa Developer Portal (available from the Tools page), and an article in the Technical Journal, 5 Things You Need to Know about LoRaWAN-based® Gateways, which may also be helpful.

If you have questions along the way, we invite you to post them in the Gateways channel of the LoRa Developer Portal Forum.

Important Notice

Contact Information

Semtech Corporation
200 Flynn Road, Camarillo, CA 93012
Phone: (805) 498-2111, Fax: (805) 498-3804
www.semtech.com