u-connectXpress software

u-blox short range stand-alone modules

User guide

Abstract

This document provides an overview of the u-connectXpress software for u-blox short range modules and describes how the products can be configured for Wi-Fi and Bluetooth use cases.

UBX-16024251 - R28
C1-Public

Document information

Title	**u-connectXpress software**	
Subtitle	u-blox short range stand-alone modules	
Document type	User guide	
Document number	UBX-16024251	
Revision and date	R28	2-Apr-2024
Disclosure restriction	C1-Public	

This document applies to the following products:

Product name	Software version
ANNA-B112	All
ANNA-B412	All
NINA-B111	All
NINA-B112	All
NINA-B221	All
NINA-B222	All
NINA-B311	All
NINA-B312	All
NINA-B316	All
NINA-B410	All
NINA-B411	All
NINA-B416	All
NINA-W131	2.0.x onwards
NINA-W132	2.0.x onwards
NINA-W151	All
NINA-W152	All
NINA-W156	3.1.0 onwards
ODIN-W260	5.0.x onwards
ODIN-W262	5.0.x onwards
ODIN-W263	7.1.0 onwards

Contents

1 Overview

This document describes how to set up and use u-blox short range stand-alone modules with u-connectXpress software. It explains the functionality of different u-blox short range stand-alone modules and includes examples that describe how to use the software in different environments with AT commands. The document is applicable for Bluetooth® Low Energy (LE), multiradio, and Wi-Fi modules.

Several u-blox short range stand-alone modules support open software variants. For more information about the available options, see the corresponding system integration manuals for u-blox short range stand-alone modules.

1.1 Product description

1.1.1 General

u-blox modules are developed for integration into a vast range of devices that demand a high level of reliability, such as those that are typically used in industrial and medical applications.

These professional grade modules operate over an extended temperature range and are approved for radio type application products in many countries. By choosing to use u-blox short range stand-alone modules, the cost and work involved in developing wireless communication solutions is significantly reduced.

Table 1 defines the most frequently used terms in this document. See also Glossary.

Concept	Definition
Host	In this document, a host refers to the device connected to a u-blox short range stand-alone module through any of the available physical interfaces. In a real application, the host is typically a microcontroller Unit (MCU) running a customer specific application.
Module	In this document, module refers to a u-blox stand-alone module. A module can also refer to a self-contained unit or item that is linked with similar units of a larger system that performs a defined task.
Peer	A connection that consists of a transmitter and one, or several, data receivers. Every transmitter and receiver in a configuration setup is referred to as a peer. A peer can either receive or send data.
Remote device	A remote device in a wireless network connecting over the Bluetooth EDR/BR, Bluetooth Low Energy, or Wi-Fi interfaces supported in the module.

Table 1: Frequently used terms

1.1.2 Multiradio and Wi-Fi modules

u-blox compact and powerful stand-alone, multiradio modules are designed for the development of Internet-of-Things (IoT) applications. These modules include embedded Bluetooth stack, Wi-Fi driver, IP stack, and an application for wireless data transfer. The wireless support includes dual-mode Bluetooth v4.0 (BR/EDR and LE) and dual-band Wi-Fi (2.4 and 5 GHz bands).

The modules support point-to-point and point-to-multipoint configurations and can accommodate concurrent Bluetooth and Wi-Fi connections. They can also operate in Wireless Multidrop™ or Extended Data Mode (EDM) for advanced multipoint capabilities. Operation in Point-to-Point Protocol (PPP) mode provides the host with a UART-based IP interface for advanced use cases. The software provides support for reduced media-independent interfaces (RMII) with micro Access Point. Some modules also have support for interfacing the module through an SPI (Serial Peripheral Interface).

1.1.3 Bluetooth Low Energy modules

u-blox Bluetooth Low Energy (LE) modules are ultra-small, high-performance, standalone Bluetooth LE modules. They are delivered with u-connectXpress software that provides support for u-blox Bluetooth LE Serial Port Service, Generic Attribute Profile (GATT) client and server, Bluetooth beacons, Near Field Communication (NFC), simultaneous Peripheral and Central roles – all configurable from a host by means of AT commands.

2 Key features

The possibility of replacing serial cables with simple wireless connections is a key feature of u-blox modules. It allows system hosts to transfer data to one another over wireless Bluetooth connections that are established between u-blox modules in Central/Peripheral configuration.

Depending on the module capabilities, data from each host is transferred to local u-blox modules over a serial UART interface, Ethernet/Reduced Media Independent Interface (RMII) or SPI. The same data is shared over the wireless link between each module.

u-blox modules can be configured to automatically establish new connections and/or accept incoming connections using AT commands. For connected hosts, this means that physical serial cables can be replaced with more convenient wireless solutions.

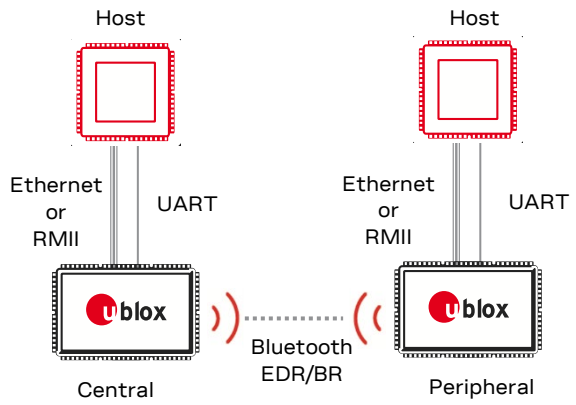


Figure 1: Bluetooth SPP connection

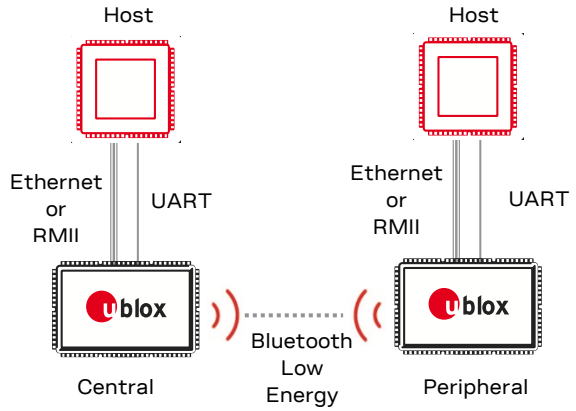


Figure 2: Bluetooth Low Energy SPS connection

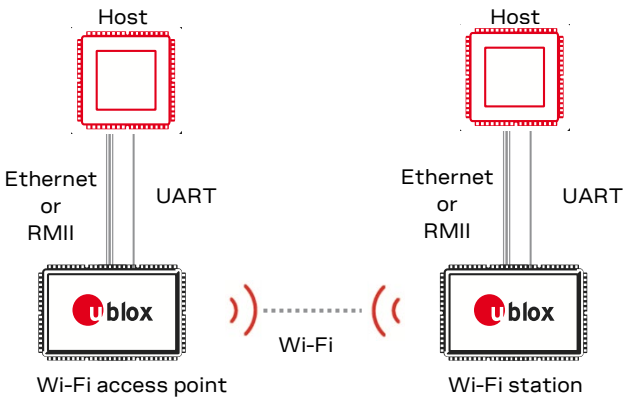


Figure 3: Wi-Fi connection

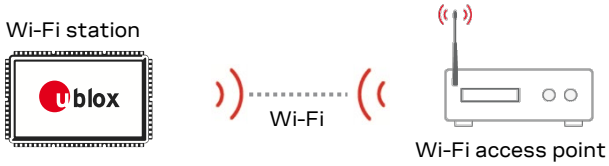


Figure 4: Wi-Fi station connection

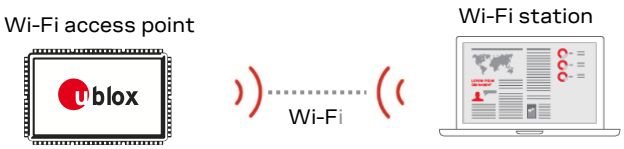


Figure 5: Wi-Fi access point connection

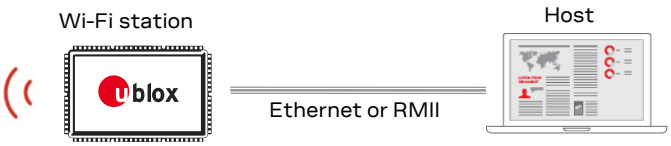


Figure 6: Ethernet connection to host and wireless Ethernet

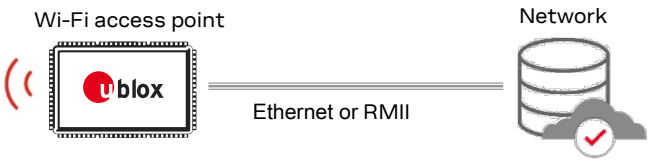


Figure 7: Ethernet connection host and wireless router

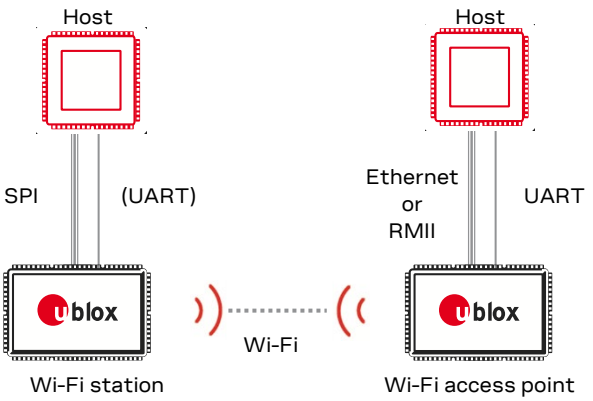


Figure 8 Example setup where a bridge configuration is used to relay data from SPI over Wi-Fi

2.1 Typical use case scenarios

2.1.1 Industrial automation

u-blox short range stand-alone modules are used in connected wireless tools for configuration and remote tracking scenarios. In these applications, any host network system can connect over Wi-Fi to any Access Point (AP) situated on the network.

In the example below, APs and mobile smart devices communicate, with hand tools in a production factory to collect operational metrics, using Bluetooth. The operational information is subsequently shared and archived in a database running on the factory's network server.

In this way, equipment configuration values, process times, and performance histories, and so on, are logged and utilized by all devices connected to the network.

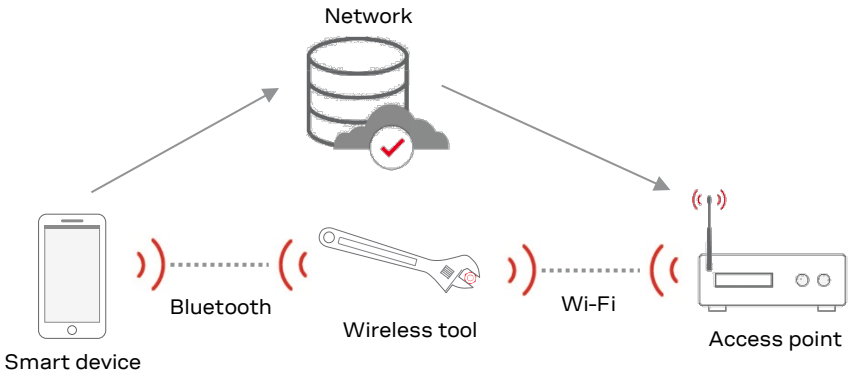


Figure 9: Industrial automation example

2.1.2 Hospital systems

u-blox short range stand-alone modules are used in various hospital systems and medical instrumentation – like infusion pumps, defibrillators, contrast injectors, multichannel EEG systems, and urology diagnostic equipment. Typically, medical staff use Bluetooth enabled barcode scanners to identify patients and track their health status. In these scenarios, Wi-Fi stations are typically used to establish hospital network connections.

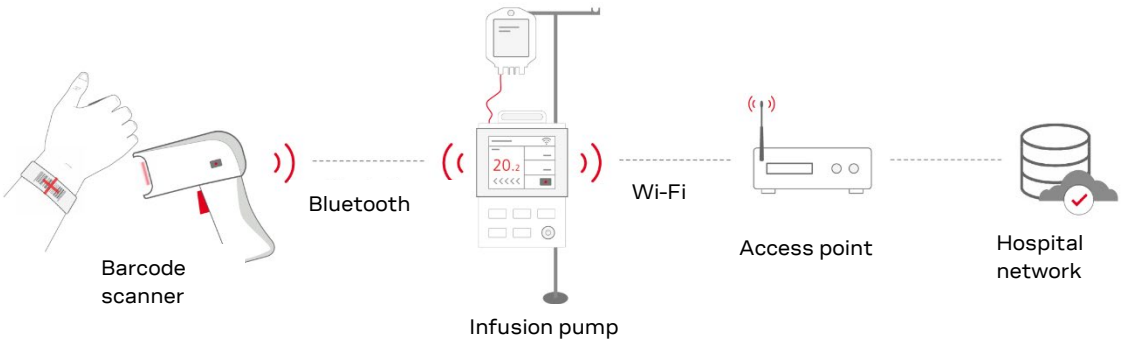


Figure 10: Hospital system example

2.1.3 Ambulance

u-blox short range stand-alone modules are implemented within ambulance instruments that check the health status of the patients in transit. In these applications, ambulance staff use smart devices, like mobile phones or tablets, through which patient data is transferred over Wi-Fi or Bluetooth to the hospital cloud. In this way, emergency hospital staff are kept well informed about the blood type, heart rate, and criticality of any incoming patients.

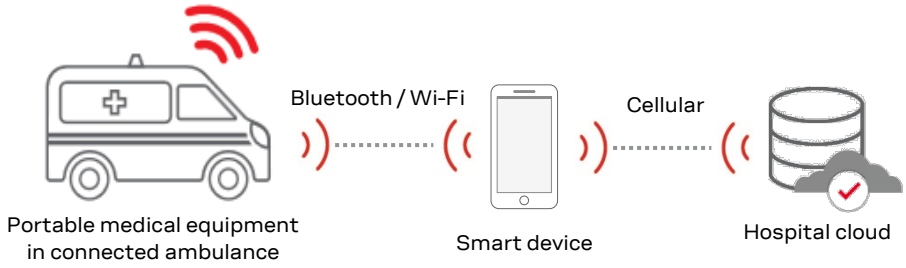


Figure 11: Ambulance example

2.1.4 Fitness

u-blox short range stand-alone modules are used in various kinds of fitness equipment like cross training equipment and exercise treadmills. During their workout, gym visitors connect to sports equipment with Bluetooth-connected smartphones. Performance metrics for any individual training pass is subsequently communicated over Wi-Fi connections to the Local Area Network (LAN).

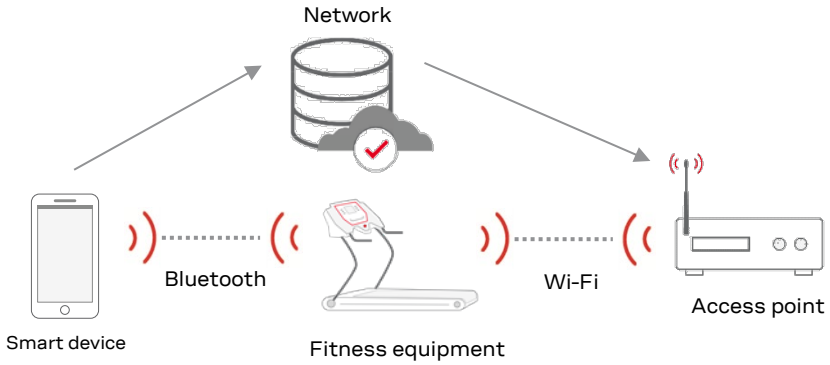


Figure 12: Fitness equipment example

3 u-connectXpress software

3.1 Software architecture

u-connectXpress software for u-blox short-range stand-alone modules makes it easy to integrate Bluetooth and Wi-Fi connectivity into new and existing products.

In several high-end modules, u-connectXpress software contains separate stacks for Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR), Bluetooth Low Energy (LE), and wireless (TCP/IP). The necessary Wi-Fi and Ethernet drivers are also included. Other module variants support different combinations of these stacks.

Figure 13 shows the logical components for high-end (ODIN-W2 and NINA-W15) modules.

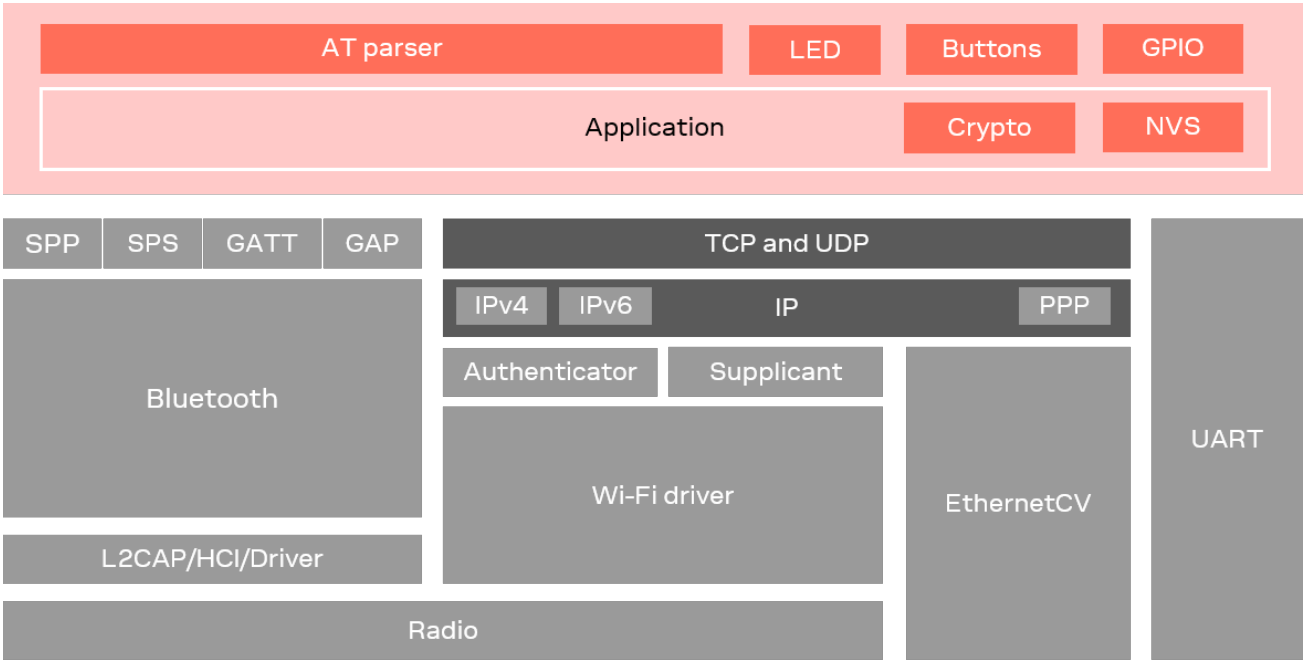


Figure 13: u-connectXpress software architecture

3.2 Operating modes

The module operates in the following modes:

- Command mode (default)
- Data mode
- Extended data mode (EDM)
- PPP mode

In addition, the module supports a number of different low-power modes, allowing power consumption optimization, independent of operating mode. See also Low power modes and Power consumption optimization.

3.2.1 Changing operating modes

u-blox modules can be configured to start in any operating mode. Once up and running, the modules can be switched between most modes – except EDM and PPP. The modes are changed with a command or escape sequence sent to the module.

Figure 15 shows how an AT command is used to switch from Command mode to either Data mode, Extended data mode, or PPP mode. It also shows how to change from data mode to command mode by sending an escape sequence over UART, or by toggling the DTR. Once the module is in Extended data mode or PPP mode, the only way to return to the command mode is to restart the module.

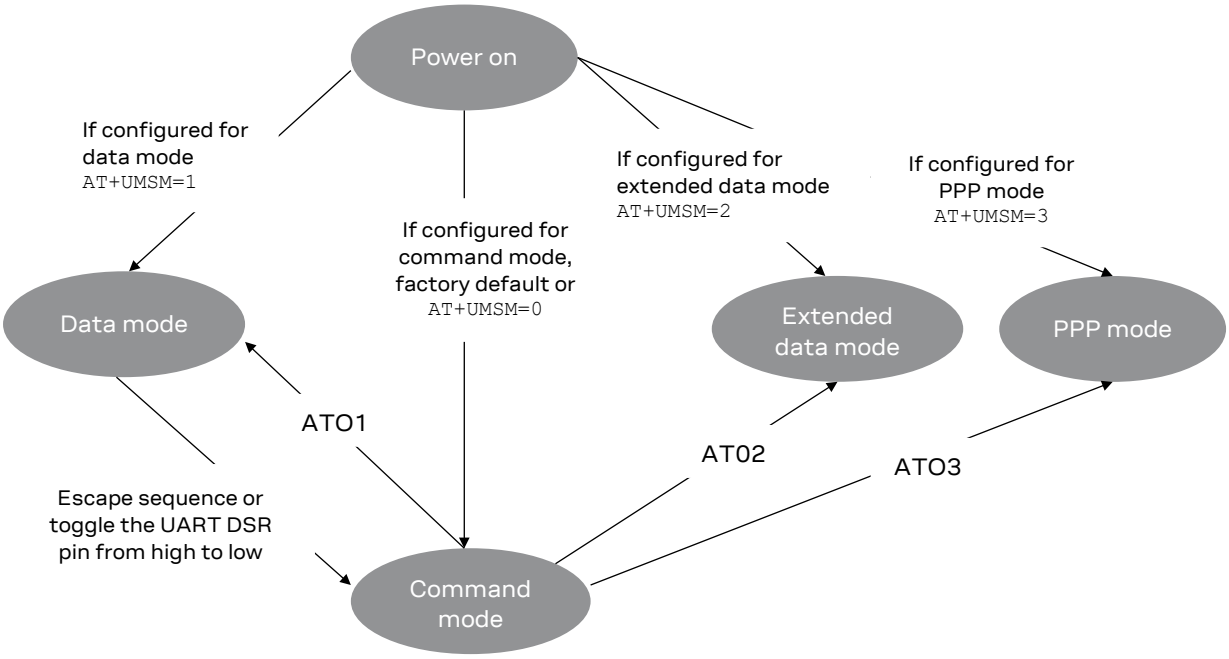


Figure 15: State diagram showing operational mode transitions

3.2.2 Command mode

The module is controlled using AT commands in (default) Command mode. In this mode the host sends control and configuration commands and indicates when data is to be sent over the UART interface. The command categories used in this mode are summarized in Table 2.

Command type	Description
Set	Configures the preferred settings for the specified command. Use of this command provides the only way to set the preferred settings in the DCE. Parameters set with this command are normally usable immediately and can be stored to the startup database using the command &W. Some configuration settings require restart of the module. Store with &W and reset with +CPWROFF.
Read	Provides the current settings of the command parameters used to find out the current command configuration.
Status	Provides current operating status of the module.
Action	Forces the DCE to print information text or execute a specific action for the command.
Configuration action	Some configuration commands require that the configuration is reset, stored, activated, or deactivated using a corresponding configuration action command.
Unsolicited result code (URC)	String messages (provided by the DCE) that are not triggered as an information text in response to a previous AT command. URCs can be output by the module at any time to inform the host of a specific event or status change. Typically, URC events occur when connections are established while disconnecting.

Table 2: Command types

Figure 16 shows the AT, OK, and URC interaction between the host and module.

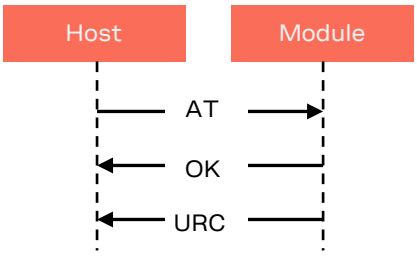


Figure 16: AT command and URC examples

For more information about all available AT commands, see the u-connectXpress AT commands manual [6].

3.2.3 Data mode

The data mode supports point-to-point and multipoint connections. In this mode, multipoint connections are supported by the Wireless Multidrop feature.

As shown in Figure 17, all local host-to-module data and data received from the remote device is transferred over the UART interfaces. Data between the local and remote modules is wirelessly communicated over Bluetooth and/or WI-FI connections. The antenna and transceiver in each module accommodate the data traffic over the air.

In data mode, user data is automatically framed and managed to accommodate the need for the wireless protocol to match its peer on the remote device. For example, the module can be configured to open a TCP server port for a remote device or connect to the TCP server port on a remote device. Once the module has connected to its peer on the remote device, data can flow between the UART interface of the module and the remote device socket. This means that the host has no need for an IP stack and can operate without any involvement in TCP handshakes, retransmissions, and so on. Similarly, all UDP, Bluetooth or Bluetooth Low Energy protocols, such as SPP or SPS, are handled by the module and not the host.

Figure 17: Data mode

It is possible to leave data mode, and return to command mode, by sending an escape sequence.

3.2.4 Extended data mode

As shown in Figure 18, the Extended data mode (EDM) allows for the individual control of each active connection. This makes it possible to transmit and receive data separately on each active connection. It also makes it clear from which remote device the data is received.

While sending and receiving data in this mode, AT commands are sent simultaneously to the module from the host. EDM is an alternative to Wireless Multidrop and is often used in more advanced multipoint scenarios, such as, allowing the host to implement an HTTP or FTP server or connect to its own FOTA server.

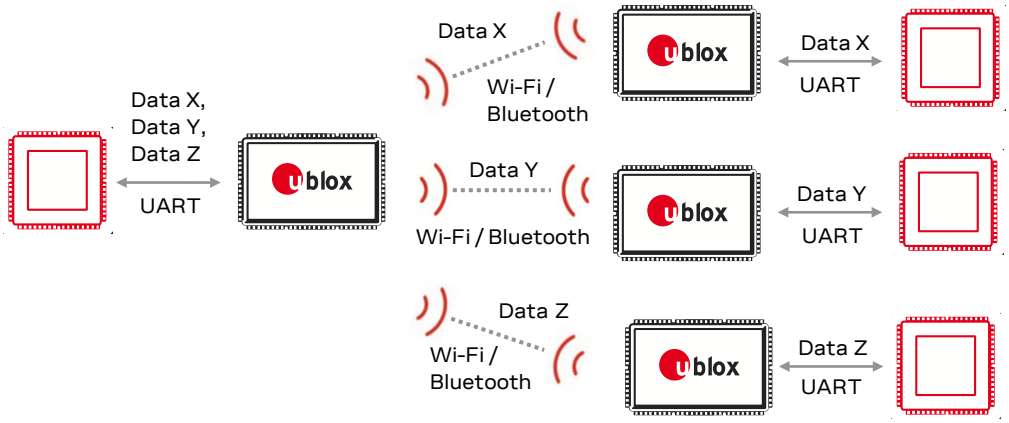


Figure 18: Extended data mode

EDM implements a method to control the module without having to explicitly enter Command mode.

As the EDM protocol is only used over the local UART interface, only "raw" data is transmitted between local and remote modules. Typically, one side of the module is configured for EDM with the other side configured for data mode.

For more information about EDM, see the u-blox Extended Data Mode protocol specification [8].

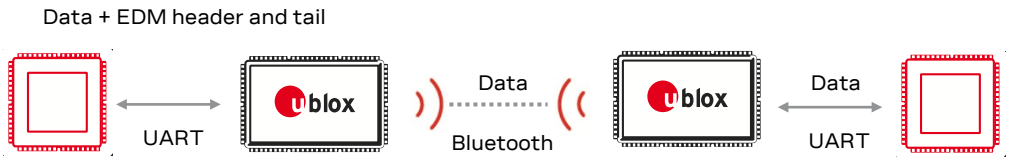


Figure 19: Data mode and extended data mode

3.2.5 PPP mode

As shown in Figure 20, PPP mode allows a host to implement an IP stack which uses a module as a network interface. PPP included in the module software supports the Server role.

In this mode, it is possible to configure an ATP server on the module that receives AT commands from the host on a specific TCP or UDP port.

To enter the Point-to-Point (PPP) mode, use the AT commands `AT03` or `AT+UMSM=3`. In this mode, data sent over the UART interface is formatted as Ethernet frames, which means that PPP connection (between the host and remote device) carries IP traffic. The host connected to the module must support the PPP client role.

Typical PPP Clients include the “dial-up modem” in Windows, and Point-to-Point Protocol daemon (pppd) in Linux.

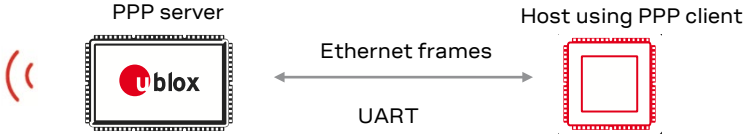


Figure 20: PPP mode to host using PPP

In PPP mode, the module operates as either a Wi-Fi Station or Wi-Fi access point.

3.3 Low power modes

The module supports several different low-power modes, allowing power consumption optimization, independent of operating mode:

- ACTIVE mode
- STANDBY mode
- SLEEP mode
- STOP mode

☞ The availability, implementation, and efficiency of each power mode varies between the different module types. For information about the power mode implementation, configuration, and optimization, see also the u-connectXpress AT commands manual [6] and module data sheet [21] [22] [23] [24] [25][26][27][28][29].

3.3.1 ACTIVE mode

When the module CPU is running at full speed, and the module is transmitting or receiving at high speed, the module is in ACTIVE mode. Whenever the module stops transmitting, it automatically leaves ACTIVE mode, and enters STANDBY mode.

3.3.2 STANDBY mode

When the module does not need full CPU and radio utilization, it automatically enters STANDBY mode to preserve power. This is the default mode after power-on.

In this mode, the radio is still powered and passively listening for incoming data, and the module is ready to accept AT-commands or data from UART, SPI and RMII.

The module may also scan, transmit beacons, advertise, accept connections, connect, etc., and keep existing connections alive in the background as well as transmit data at “low speed”.

However, power to the CPU and the radio may be automatically reduced, and the module may even automatically enter SLEEP mode, during “short” periods that do not affect the radio performance. In that case, u-connectXpress automatically restores power and returns from SLEEP mode when needed.

Configuration options such as Wi-Fi DTIM and listen intervals, as well as Bluetooth advertising periods and Automatic Frequency Adaption (AFA) have a direct effect on the overall power consumption of the STANDBY mode.

Whenever needed, the module seamlessly enters ACTIVE mode until that is no longer needed.

3.3.3 SLEEP mode

During SLEEP mode, the CPU may be temporarily halted, and the radio may be temporarily powered down – listening only occasionally for incoming data or control packets.

The module may automatically enter SLEEP mode, during “short” periods that do not affect the radio performance. In that case, u-connectXpress automatically restores power and return from SLEEP mode when needed.

The host may also choose to disable the UART when DTR control has been set to `AT&D3`. In these circumstances, it is not possible to either transmit data over radio or read incoming data. As a result, application-level protocols that require response from the host may time-out unless the UART is re-enabled to read and act on incoming data at suitable intervals.

Unless configured with suitable TCP keep-alive or peer reconnect timeouts, remote peers may ultimately drop the connection due to inactivity timeout.

However, RAM and radio connection state is retained, so Wi-Fi and Bluetooth connections are retained.

The transition-time from SLEEP mode to STANDBY or ACTIVE mode is “short” and – other that the time taken to enable the UART – does not affect the timing on the host.

☞ The availability, implementation and efficiency of the SLEEP mode highly depends on several things, such as if external low-power clocks/oscillators are required, if the module is configured as Wi-Fi AP or Station, Bluetooth Central or Peripheral etc.

3.3.4 STOP mode

The host may force the module into STOP mode. u-connectXpress never puts the module into STOP mode automatically.

In STOP mode, the CPU and radio are completely powered down and all connections are dropped. Except for the settings stored to profile or NVRAM, the RAM may be cleared as well.

The RTC, if available, may continue to be powered.

The host may re-start the module by either toggling DTR (when DTR control has been set to AT&D4), a GPIO, or as a result of a pre-configured timeout.

The transition-time from STOP to STANDBY can be up to 5 s, depending on module.

☞ The availability, implementation and efficiency of the STOP mode highly depends on the actual module hardware, if external low-power clocks/oscillators are required, etc.

3.4 System control signals

Module configuration and control is determined by the position of switches, and LED indicators reflect the operating mode and connection status of the module.

3.4.1 Switches and input signals

A module running u-connectXpress can be reset to the factory default by running the command `AT+UFACTORY`, or by using the system control signals.

In general, the following input signals are used to control the system:

- **RESET_N** resets the system.
- If **SWITCH_2** (on ANNA-B1, ANNA-B4, NINA-B1, NINA-B3 and NINA-B4), or alternatively **SWITCH_1** (on NINA-B2, NINA-W13, NINA-W15 and ODIN-W2), is driven low during start up, the serial UART settings are restored to their default values.
- If both **SWITCH_1** and **SWITCH_2** are driven low during start up, the system enters the bootloader mode.
- If both **SWITCH_1** and **SWITCH_2** are driven low during start up and continue to be held low for a duration of 10 seconds, the system exits the bootloader mode and restores all settings to their factory default values.

☞ Note that **SWITCH_1** and **SWITCH_2** are called **SWITCH_0** and **SWITCH_1** on ODIN-W2 modules.

☞ If you are using a u-blox evaluation board, you can restore the factory default settings by pressing buttons SW1 + SW2 for a period of 10 seconds after the board reset.

Other important input signals and functions are:

- **SWITCH_2** can be used to open a Bluetooth LE connection with a Peripheral device.
- **DTR** changes the operating mode. For information about defining the mode controlled by DTR, see the `AT&D` command description in the u-connectXpress AT commands manual [6].

- **CTS** and **RTS** are used for handshake over UART. It is strongly advised that the CTS/RTS handshake is implemented on the host. Otherwise, data may be lost.
- **SPI_CS** and **SPI_CLK** are used to activate the u-connectXpress SPI control protocol during startup. See also the Communicating with a u-blox module over SPI bus application note [32].
- **RMII_CLK** is used to activate support for RMII in some modules.

☞ Note that DTR refers to the DTR **output** signal of the host, which is connected to the DSR **input** pin on the module.

For more information about any product specifics, see the appropriate EVK user guide, system integration manual, and data sheet for your module.

3.4.2 LED and output signal indicators

RED, **GREEN,** and **BLUE** pins signal the operating mode and connection status of the module. These active-low pins are typically routed to an RGB LED to provide a visual status of the module.

The various color combinations used to indicate the module mode and status are shown below.

Mode	Status	RGB LED color	GREEN	BLUE	RED
Data mode	IDLE	Green	LOW	HIGH	HIGH
Command mode	IDLE	Orange	LOW	HIGH	LOW
Data mode, Command mode	CONNECTING*	Purple	HIGH	LOW	LOW
Data mode, Command mode	CONNECTED*	Blue	HIGH	LOW	HIGH

* LED flashes on data activity

Table 3: Typical operating mode and status indication

For further information about the LED indicators, see the respective product data sheet.

The **DSR** signal can be configured to indicate peer connection status. For details on how to define the status indicated by the DSR signal, see the documentation for the `AT&S` command.

☞ Note that DSR refers to the DSR **input** signal of the host, connected to the DTR **output** pin on the module.

The **DRDY** signal indicates availability of data that an SPI master reads using the u-connectXpress SPI control protocol. See also the Communicating with a u-blox module over SPI bus application note [32].

3.4.3 Escape sequence

The default escape character for u-connectXpress software is “+” (ASCII value 43). The escape sequence is triggered by the following sequence:

1. Silence 1 second
2. `+++`
3. Silence 1 second

`+++` must be sent within 200 ms, which makes it difficult to enter the escape sequence manually using a terminal window. It is recommended that you “paste” the characters to ensure that they are sent as fast as possible. You can also enter the command mode by toggling the **UART DTR** pin from high to low.

3.5 Client and server roles

A server provides a function or service to one or many clients that initiate requests for such services. For the module, this service is typically access to a data channel. The normal case is that the client “wants the data” and the server “has the data”.

3.5.1 Wi-Fi Access Point and station

The Wi-Fi station is a client that connects to the Access Point. The Access Point then broadcasts beacons and allows stations to connect; the Access Point can handle many stations.

3.5.2 Bluetooth BR/EDR Central and Peripheral

A Bluetooth BR/EDR device supports up to seven parallel Bluetooth connections – this is called multipoint. By default, the client becomes the Central and the server becomes the Peripheral. If a server wants to support multiple connections and still wants to have a Piconet for best performance, the server must request a Central/Peripheral switch for every incoming connection.

3.5.3 Bluetooth Low Energy Central and Peripheral roles

A Bluetooth Low Energy (LE) device either supports the Central, Peripheral, or both roles. The central is the client and makes connection to the Peripheral, which is a server. The Peripheral is typically a battery powered device like a sensor, and the Central, is often a smartphone or a computer.

3.6 Peers

3.6.1 Introduction

A connection consists of a sender and one or several receivers of data. Every sender and receiver in a setup is referred to as a peer. A peer can either receive or send data. There are two kinds of peers:

- Local peer
- Remote peer

The local peer is synonymous with the UART. The remote peer is another device or the broadcast range on the network. Several remote peers can be defined in a Wireless Multidrop scenario.

A remote peer is addressed using a Uniform Resource Locator (URL). These URLs are strings representing nodes on the Internet or on a local net. It is the same addressing technology as used in a web browser. For more information about URLs, see the Bluetooth Assigned Numbers, specification [7].

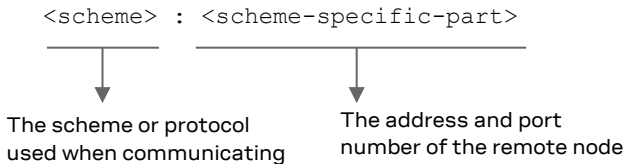


Figure 21: URL example

For example, a web server on the Internet can be assigned with https://www.u-blox.com as the address. This tells the browser to use the HTTPS protocol and connect to the node at address https://www.u-blox.com. A similar addressing scheme is used by the module to pinpoint the remote peer. The scheme is not "https", but the node addressing is identical.

Available schemes:

- `tcp`: TCP connection, including TLS
- `http-tcp`: HTTP over TCP, including HTTPS
- `udp`: UDP connection, broadcast capabilities
- `spp`: Bluetooth Serial Port Profile
- `dun`: Bluetooth Dial Up Networking
- `sps`: Bluetooth Low Energy u-blox Serial Port Service

- `mqtt`: MQTT over TCP, including TLS support for encryption and authentication
- `spi`: SPI interface

Syntax:

- tcp/udp/mqtt/http-tcp: `<scheme>://<ipaddress>:<portnumber>/[?<option>{&<option>}]`
- spp/dun/sps: `<scheme>://<bluetooth_address>/[?<option>{&<option>}]`
- spi: `<scheme>://<spi_interface>/[?<option>{&<option>}]`
- option: `key=value` or `key=%%n`

Remarks:

- The IP address can be either a numeric IP address or a host and domain name that can be resolved using the configured DNS servers
- The options are scheme-specific
- Bluetooth devices can have public or different variants of random addresses. When addressing Bluetooth devices in u-connectXpress, this is indicated by adding an optional "p" or "r" as suffix on the address. For example, `01A0F7101C08p`. If no suffix is present, the address defaults to a public address.

Example URLs:

- `tcp://10.0.0.9:5003`
- `tcp://echo.u-blox.com:7`
- `tcp://www.u-blox.com:443/?encr=1`
- `udp://192.168.0.42:6809`
- `spp://0012f3000001`
- `mqtt://test.mosquitto.org:1883/?pt=u-blox/mytopic&st=u-blox/#&mode=1`

3.6.2 TCP peer

A TCP peer is the same as a TCP socket. When a TCP peer is connected, data can flow in both the directions irrelevant of whether the peer is a server or a client. To optimize the TCP link for short latency, the `<flush_tx=1>` can be specified in the URL – although this is not needed in most cases.

The TCP keep alive timeout can also be set using `<keepalive>` in the URL.

TCP connections can optionally be encrypted and/or authenticated using TLS to allow end-to-end encryption between the peer and the module.

URL	Address and option
`tcp://192.168.0.1:8080`	Using IPv4 address
`tcp://192.168.0.1:8080/?flush_tx=1`	Using short latency
`tcp://192.168.0.1:8080/?keepalive=5000+1000+5`	Using keep alive for 10 seconds
`tcp://[FE80::7AA5:4FF:FE2F:5F01]:8080`	Using a IPv6 address
`tcp://192.168.0.1:8080/?encr=1`	Using TLS encryption
`tcp://192.168.0.1:8080/?ca=ca.crt&cert=client.crt &privKey=client.key`	Using TLS 2-way authentication

Table 4: Example to connect to port 8080 with addressing and options

3.6.3 UDP peer

A UDP peer is the same as a UDP socket. For the UDP peer, the behavior differs for servers and clients. A server accepts data from any IP address to the activated port number.

A client can be used to send data to a specified address. To listen on a different port than the remote port, specify the `<local_port>` in the URL.

☞ The UPD connection is setup in one direction only.

URL
udp://192.168.0.1:8080/?local_port=8081

Table 5: Example to send on port 8080 and receive on port 8081

3.6.4 SPP peer

The SPP peer is the Bluetooth BR/EDR Serial Port Profile.

A client can be used to send and receive data to and from a specified address. To connect to a different port other than the remote port, specify the same in the URL.

URL
spp://112233AABBCC
spp://112233AABBCC/?port=1

Table 6: Example to connect SPP to remote device

3.6.5 SPS peer

The SPS peer is the Bluetooth Low Energy Serial Port Service.

A client can be used to send and receive data to and from a specified address.

URL
sps://112233AABBCC
sps://112233AABBCC/?role=p

Table 7: Example to connect SPS to remote device

☞ When a u-blox Bluetooth LE module makes an SPS connection to another u-blox Bluetooth LE module, you get two connection handles: one connection handle and one peer handle. The ACL event (+UUBTACLC) gives you a connection handle that is used with the GATT ACL protocol.

☞ Use the connection handle to perform operations on GATT characteristics. The peer handle is related to a u-blox stream, in this case the SPS protocol, which is built on top of GATT. See also the SPS Protocol Specification [31].

3.6.6 MQTT peer

An MQTT peer is a connection to an MQTT broker using the MQTT protocol with optional encryption and authentication. When an MQTT peer is connected, the data between the module and the host is either sent directly to the specified topic, and received from the subscribed topics, or sent and received as MQTT-SN. This allows the module to act either as an MQTT gateway or MQTT-SN gateway depending on the URL.

☞ If an MQTT stream is configured without the “pt” and “st” keys, it is instead configured as an MQTT-SN stream. If any of the “pt” or “st” keys are used, the stream is configured as an MQTT client stream.

☞ For the subscribe topic, the multi-level character, ‘#’, is supported (as in st=ubx/test/#). The single-level character - ‘+’ is not supported.

☞ Occasionally, the option values can be very long and/or contain characters that interfere with the keys defined by u-connectXpress. In those cases, use AT+UDUV to define the values to be used in the URL, and use the key=%%n-syntax in the URL instead of key=value for the affected option.

URL	Module role
mqtt://test.mosquitto.org:1883/?pt=ubx/mytopic	MQTT gateway, publish ubx/mytopic
mqtt://test.mosquitto.org:1883/?st=ubx/#&mode=1	MQTT gateway, subscribe to all under ubx
mqtt://test.mosquitto.org:1883/?st=ubx/mytopic	MQTT gateway, subscribe to ubx/mytopic only

`mqtt://test.mosquitto.org:1883/?maxSnClients=24`	MQTT-SN gateway, with up to 24 MQTT-SN clients
`mqtt://test.mosquitto.org:1883/?pt=ubx/mytopic&password=%%0`	MQTT gateway, publish ubx/mytopic, authenticate with password previously set using `AT+UDUV`

Table 8: Example to connect to an MQTT broker

More information about how to use MQTT and MQTT-SN is found in the u-connectXpress MQTT application note [19].

3.6.7 HTTP-TCP peer

A HTTP-TCP peer defines how to connect to a HTTP/HTTPS server, optionally using TLS. The HTTP-TCP peer supports all keys supported by a TCP peer, plus setting HTTP timeout.

However, the actual connection is deferred to until the HTTP/HTTPS path is defined. Hence, it is possible to issue several requests to the same HTTP/HTTPS server, without having to create and disconnect peers for every request.

URL	Address and option
`http-tcp://192.168.0.1:443/?encr=1`	Using TLS encryption

Table 9: Example to connect to port 443 using HTTPS over TLS

3.6.8 SPI peer

An SPI peer allows the module to act as an SPI slave capable of forwarding data to/from another stream, as an alternative to enabling AT over SPI during module startup.

URL	Address and option
`spi://spi0/?cs=32&sclk=31&miso=36&mosi=35&mode=3&drdy=25&size=720&proto=3`	SPI slave with PDU size 720, SW protocol 3

Table 10: Example to enable data stream on default SPI pins

3.7 ODIN-W2 Wi-Fi roaming

The roaming functionality supported in ODIN-W2 allows it to move between several Wi-Fi Access Points (AP) that share the same Service Set Identifier (SSID) – without losing network connection. This functionality also makes it possible for the module to move in and out of the Wi-Fi network range of any AP without losing data. In these circumstances the module repeatedly tries to send data until the network connection is restored.

☞ In both scenarios, network connection is lost if the module remains outside of the network range for any extended period of time. The exact time-out period for network connection is dependent on the chosen application protocol.

Wi-Fi Roaming in ODIN-W2 supports 802.11r with Pairwise Master Key caching (PMK) or Opportunistic Key caching (OKC).

Roaming behavior in ODIN-W2 is designed to:

- Monitor Received Signal Strength Indicators (RSSI) to direct roaming to the most suitably located Access Points
- Use the RSSI value to decide when background scanning is performed

3.7.1 Good RSSI scan and discovery

In most environments, it is desirable to configure ODIN-W2 so that it always connects to the AP with the best signal strength in the local network. But, in some situations this configuration has some drawbacks.

Because of the extra time it takes to discover all APs across several different channels, it can be appropriate for the module to connect to an AP that has only an adequate signal strength – but in a much shorter time. Consequently, some variable acceptance of weaker signal strength can expedite scanning and reduce the risk of data drops between AP nodes with faster connection times.

To accommodate both scenarios, the value of the `Good RSSI` parameter tag is configured in the range -128 to 0 dBm (default=-55 dBm).

You use the following commands to configure the appropriate roaming behavior:

- `AT+UWCFG[=5, 0]` configures the module to always connect to the AP with the highest signal strength (RSSI).
- `AT+UWCFG[=5, -128]` configures the module to always connect to the first AP that meets the signal strength (RSSI).

3.7.2 Disable roaming and timeout

To completely disable roaming set t `Slow scan sleep timeout` and `Fast scan sleep timeout` to zero, using `AT+UWCFG=7,0` and `AT+UWCFG=8,0`.

3.7.3 Bad area

To avoid fast switching when all APs are in a bad area, the previously connected AP is blacklisted for 5 seconds. At this time it is this time configurable with `AT+UWCFG=9,<timeout_in_seconds>`.

The threshold for the RSSI to trigger a roaming (Bad RSSI) is -70 by default. In some use cases, this value is too high; any value between -75 to -80 could be a better choice to prevent roaming too often. This value should be carefully selected and changed by the user for optimal performance. The roaming threshold can be changed by the Bad RSSI value `AT+UWCFG=6,<RSSI Value>`.

3.7.4 Roaming example

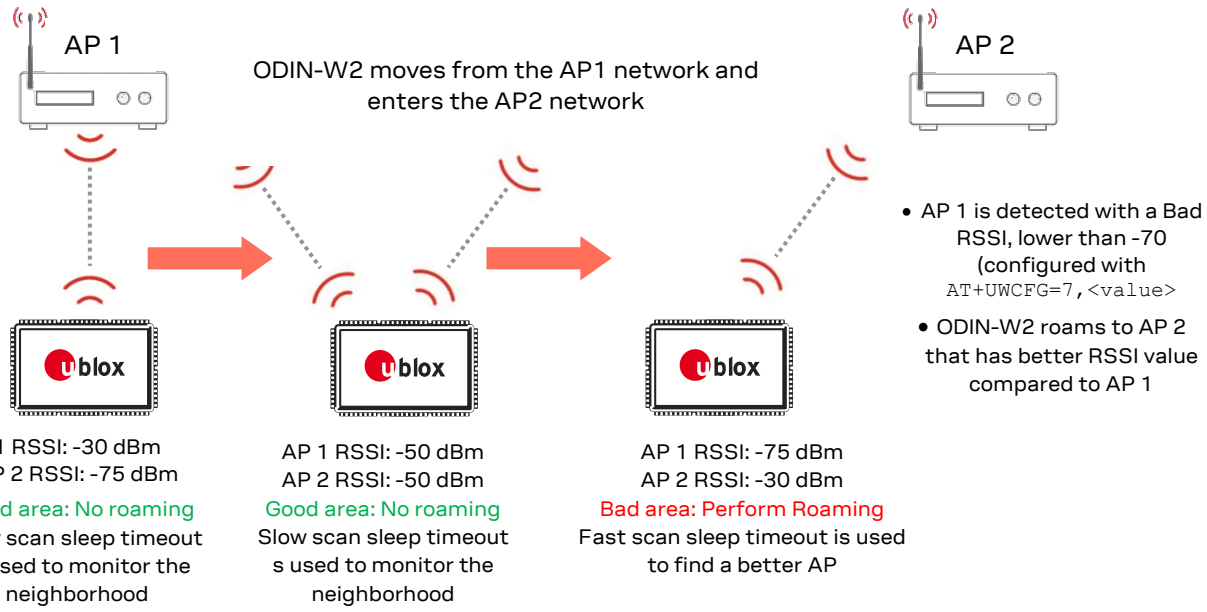


Figure 22: Basic roaming behavior

3.8 Bridge functionality

It is possible to bridge the following four different network interfaces:

- Wi-Fi Station
- Wi-Fi Access Point
- Ethernet
- Bluetooth PAN

The bridge is on Ethernet level but also bridges the IP traffic including DHCP and broadcasts packets.

The bridge interface supports static IP address and DHCP server and can (if on the same subnet) be accessed on the network.

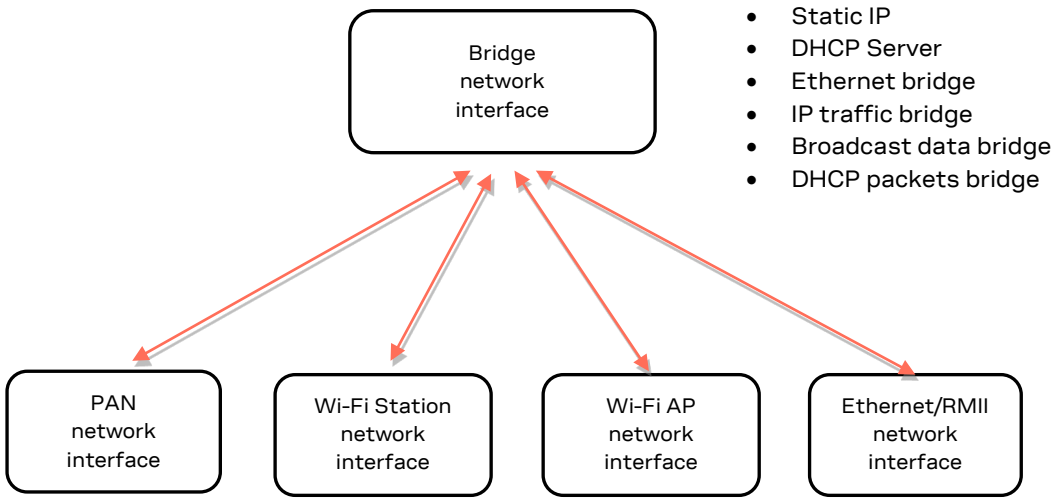


Figure 23: Bridge functionality

One bridge function is to bridge the Wi-Fi AP with the Ethernet/RMII network interface; in the current example, the Ethernet network interface is connected using a PHY to a network that has the DHCP server.

The network interface IDs include:

- Wi-Fi Station
- Wi-Fi Access Point
- Ethernet
- Reserved (do not use)
- Bluetooth PAN

It is advisable to setup the bridge before activating any network interfaces.

Use the following configuration to setup the bridge with Wi-Fi Access Point and the Ethernet (using PHY) without a DHCP server.

3.8.1 Example: a bridge configuration without the DHCP server

	Instructions	AT command
1	Bridge the Wi-Fi Access Point and the Ethernet interface (Layer-2 routing).	`AT+UBRGC=0,1,2,3`
2	Activate the Bridge.	`AT+UBRGCA=0,3`

It is also possible to bridge a single network interface, such as Ethernet. The bridge can then be used to enable the DHCP server on the Ethernet interface. See also Other use cases.

When the Bridge interface is activated, the interfaces that are used do not use or receive any IP address. To access the device the IP address on the bridge interface must be set.

3.8.2 Example: DHCP server on bridge interface between Wi-Fi access point and Ethernet interface

	Instructions	AT command
1	Use the Bridge between the Wi-Fi Access Point and Ethernet.	AT+UBRGC=0,1,2,3
2	Use static IP address.	AT+UBRGC=0,100,1
3	The IP address.	AT+UBRGC=0,101,192.168.0.100
4	Set the network mask.	AT+UBRGC=0,102,255.255.255.0
5	Set the gateway.	AT+UBRGC=0,103,192.168.2.1
6	Enable the DHCP server on the Bridge.	AT+UBRGC=0,106,1
7	Activate the Bridge.	AT+UBRGCA=0,3
8	Activate the Ethernet interface using default settings.	AT+UETHCA=3
9	Set the name of the network (SSID).	AT+UWAPC=0,2,"myssid"
10	Set the channel for the Access Point.	AT+UWAPC=0,4,6
11	Enable WPA2 security	AT+UWAPC=0,5,2,2
12	Set the password	AT+UWAPC=0,8,"mypassword"
12	Activate the Access Point.	AT+UWAPCA=0,3

3.9 IP forwarding

It is possible to configure two different interfaces with different network ranges for example, 192.168.0.1 on the Access Point interface and 10.0.0.1 on the Ethernet interface. The packets received on one network interface are forwarded to the other interface. In this case, other devices on the Ethernet network should use static address other than 10.0.0.1 and use the 10.0.0.1 as the Gateway address. The devices connected to the Access Point are accessible from the Ethernet interface using the 192.168.0.1xx address space.

```
AT+UETHC=100,1
AT+UETHC=101,10.0.0.1
AT+UETHC=102,255.255.255.0
AT+UETHC=103,10.0.0.1
AT+UETHC=1,0
AT+UETHCA=3

AT+UWAPC=0,100,1
AT+UWAPC=0,101,192.168.0.1
AT+UWAPC=0,102,255.255.255.0
AT+UWAPC=0,103,192.168.0.1
AT+UWAPC=0,104,0.0.0.0
AT+UWAPC=0,105,0.0.0.0
AT+UWAPC=0,106,1
AT+UWAPC=0,2,"myssid"
AT+UWAPC=0,4,1
AT+UWAPC=0,5,2,2
AT+UWAPC=0,8,"mypassword"
AT+UWAPCA=0,3
```

3.10 Bind functionality

It is possible to route data (Layer-4 routing) between different peers using the Bind command. The Bind functionality is bidirectional and is active if both the links are connected; the routing works in AT mode and there is no need to enter data mode to start the Bind functionality.

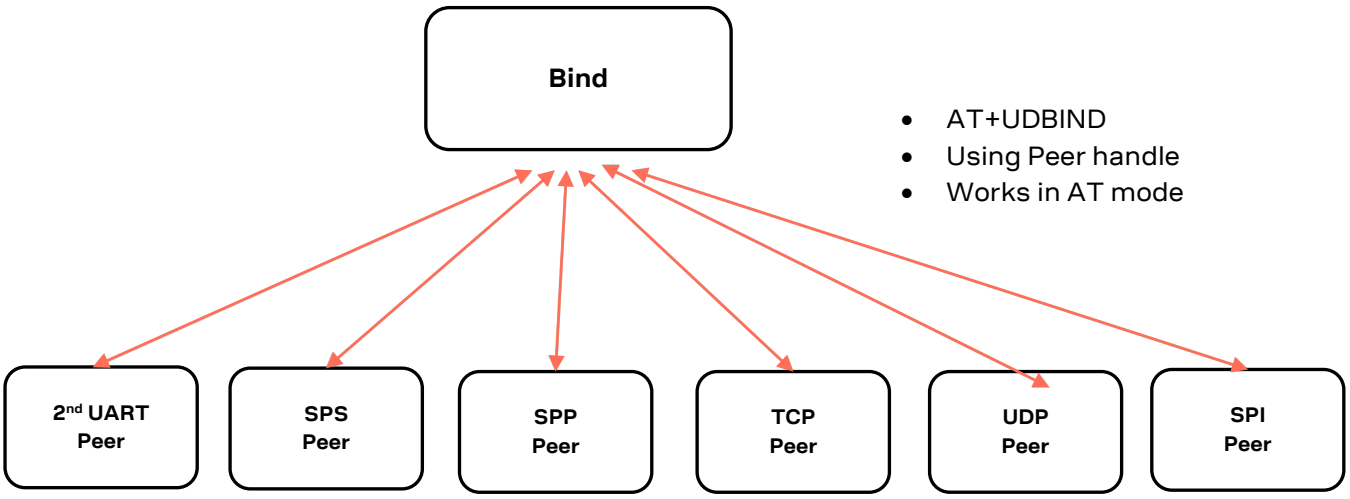


Figure 24: Bind functionality

The following different peers can (depending on module capability) be used in Bind:

- SPS
- SPP
- TCP
- UDP
- SPI
- UART (secondary)

Example: Route data between the SPS and TCP

```
AT+UDCP="tcp://echo.u-blox.com:7"
+UDCP:1
OK
+UUDPC:1,2,0,172.20.10.2,49153,195.34.89.241,7

+UUBTACLC:0,0,4888F5181AA9r
+UUDPC:2,1,4,4888F5181AA9r,20
AT+UDBIND=1,2
+UDBIND:29,28
OK
```

Example: Make a repeater between two Bluetooth SPP connections

☞ Make sure that the `AT+UBTCFG` for the number of connections is used before making more than one connection.

```
AT+UDCP="spp://48BF6B51D0C6p"
+UDCP:3
OK
+UUDPC:3,1,1,48BF6B51D0C6p,669
AT+UDCP="spp://48BF6B51F98p"
+UDCP:4
OK
+UUDPC:4,1,1,48BF6B51F98p,669
AT+UDBIND=3,4
+UDBIND:31,30
OK
```

3.11 MQTT

It is possible to configure the gateway as an MQTT-SN Gateway. This is intended for end devices that do not support TCP/TLS directly, but instead connect to the gateway using a serial connection like Bluetooth LE SPS or Bluetooth SPP. As shown in Figure 25, the devices communicate with the gateway using the MQTT-SN protocol. See also Use case 2: MQTT-SN gateway.

It is also possible to configure the gateway as an MQTT Client Gateway. As shown in Figure 25, the host of the gateway can then transmit or receive transparent MQTT data directly over the UART. See also Use case 3: MQTT client gateway.

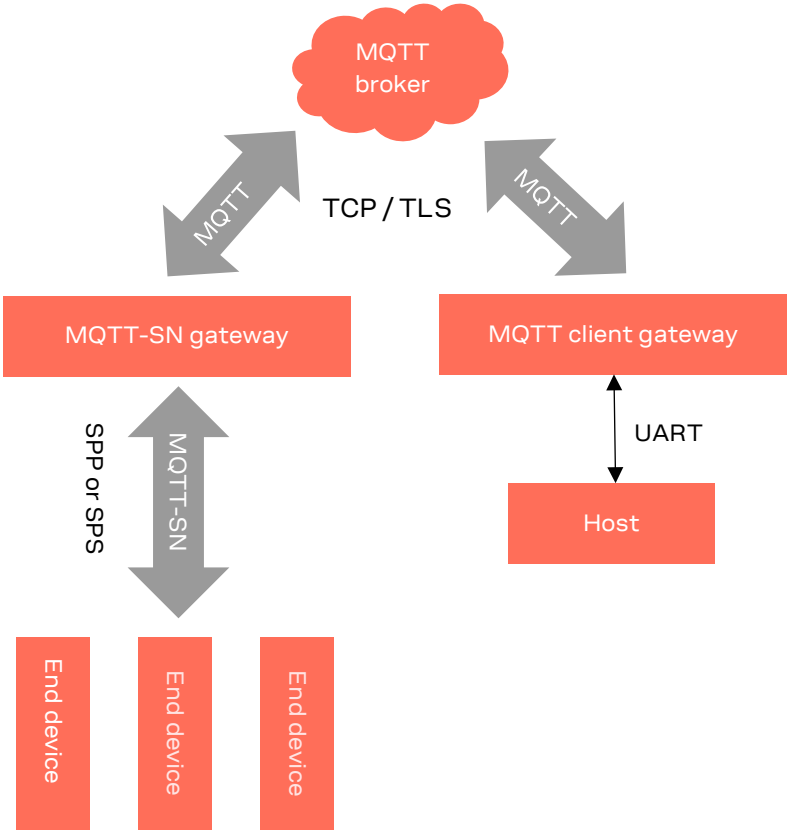


Figure 25: Device communication with the gateway using the MQTT-SN protocol

3.12 IoT cloud connectivity

It is possible to connect to popular cloud services like Amazon, Azure and IBM using TLS and MQTT. In some cases, some of these services can be connected to simultaneously. For information describing the setup and configuration of these services, see the u-connectXpress IoT Cloud connectivity application note [20].

aws qualified device

Built-in HTTP, HTTPS and NTP clients simplify authentication and communication with any remote server.

3.13 Security

To prevent unauthorized access to connected devices over wireless networks, u-connectXpress software includes separate security mechanisms for Wi-Fi, Bluetooth, and Transport Layer.

3.13.1 Wi-Fi security

This section only covers the different topics very briefly, for more detailed information please see the u-connectXpress Wi-Fi security application note [34].

u-blox standalone modules support several variants of Wi-Fi security, they are described in the following chapters.

3.13.1.1 Wi-Fi security combinations

The matrix below shows the valid combinations of supported Wi-Fi security modes.

Table 11 describes the Wi-Fi security modes and related encryption security protocols supported by u-blox short range standalone modules.

Wi-Fi security mode	Encryption security protocols				
	Unencrypted	TKIP	AES/CCMP	PMF	Key method
Open (no security)	Valid	-	-		
WPA2-PSK (Personal)	-		Valid	Supported	PSK
WPA3-PSK (Personal)	-		Valid	Optional	SAE
802.1X (Enterprise) EAP-TLS, PEAP, LEAP	-	Valid (only for station)	Valid (only for station)		

Table 11: Wi-Fi security support in u-blox short range modules

☞ WEP and TKIP are considered as unsecure. The WEP is deprecated in the 802.11i specification.

Wi-Fi Protected Access 2 (WPA2), also known as WPA-Personal or 802.11i, is the most common security setting for Wi-Fi networks. The WPA2 has replaced WPA.

If WPA/WPA2/WPA3 is used by the Wi-Fi Station; the WPA2/WAP3 with AES/CCMP encryption is used, if supported by the Access Point. If not, a WPA with TKIP encryption is used.

☞ Note that all u-blox standalone modules (and all SW versions) do not support WPA3.

☞ It is not possible to have the WPA with AES/CCMP encryption or WPA2 with TKIP.

3.13.1.2 Key management

3.13.1.2.1 WPA PSK

In WPA/WPA2 PSK, AES/CCMP is used for unicast packets and TKIP is used for broadcast packets using either the pre-shared key (that is, the hexadecimal string) or the password (plain-text) commonly referred to as "WPA-PSK" and "WPA-PWD". Whenever you change the password, you need to Deactivate and Activate for the settings to take effect. If you choose to enter a password (not a hexadecimal string), the module takes slightly longer during activation or boot after this change, in order to deduce the real key from the password.

The hexadecimal strings are given byte-by-byte. Each hexadecimal coded byte is prepended with the string escape character “\”. For example: "\AF\11\12\4C\00\FF\0A\6D".

3.13.1.2.2 Enterprise security

Enterprise security is the common name for all the methods that use 802.1X to authenticate with a backend RADIUS server. When using enterprise security, some credentials must be stored in the module; typical credentials include username, domain name, and password or certificate.

The 802.1X authentication leads to the exchange of a Master Session Key that can be used either for WEP encryption or WPA based security.

The username, password and the domain for the RADIUS server should be entered by the user.

If you wish to use enterprise security (LEAP, PEAP, or EAP-TLS) as the authentication algorithm, ensure that your access point supports it. Not all the access points support enterprise security.

3.13.1.2.3 Certificate management

EAP-TLS uses certificates and keys; these files are stored in the internal storage in the module. The certificate and the private key, which may be encrypted, must be selected thus making an EAP-TLS connection.

During a certificate request, there is a signing procedure. This is the equivalent of calculation A^E modulus N, where A, E and N are values in the size of the certificate. That is, this is an operation that is designed to take a lot of processing time.

The processing time varies for different certificates. Listed below is an example of the processing time required by different certificate sizes.

☞ The processing time is exponentially increased whenever the certificate size doubles

- 512 bit - 250 ms
- 1024 bit - 500 ms
- 2048 bit - 1000 ms
- 4096 bit - 3000 ms

3.13.1.2.4 PEM format certificates

PEM files are essentially a group of base64 encoded DER certificates and keys with additional metadata. This allows the stored keys to be encrypted within the PEM file. If the PEM file contains more than one certificate, the complete order is determined, and the certificates are sent as a certificate chain.

- Keys may be encrypted using either: `DES(DES-CBC)` or `3DES (DES-EDE3-CBC)`
- Encrypted keys must be contained within:
  `-----BEGIN RSA PRIVATE KEY-----"` and `"-----END RSA PRIVATE KEY-----`
- Clear text keys should be contained within:
  `-----BEGIN PRIVATE KEY-----"` and `"-----END PRIVATE KEY-----`
- Certificates should be contained within:
  `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----`
- Certificate encryption is not supported
- The order of certificates within the PEM file is not important; the certificates are sorted so that the order in the TLS packet is correct.
- Only one key should be present within a PEM file
- All certificates within a PEM file must belong to the same (straight) chain

Example to generate a self-signed certificate (should be used only for evaluation):

```
$ openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:2048 -keyout privateKey.key -out
certificate.crt
```

Convert a PKCS#12 file (`.pfx .p12`) containing a private key and certificates to PEM:

```
# Output the entire certificate chain to cert_chain.pem.
openssl.exe pkcs12 -in certificate.pfx -out cert_chain.pem -nokeys

# First convert the pfx file to PEM, then reformat the PEM file with the RSA module to get
a compatible PEM encrypted key.
openssl.exe pkcs12 -in certificate.pfx -out key.pem -nocerts
```

3.13.1.2.5 PER to DER format conversion

DER is the raw format of certificate that cannot be encrypted and can only hold a single certificate or key.

Example of converting a certificate in PEM format (.crt .cer .pem), to DER format:

```
$ openssl x509 -outform der -in certificate.pem -out certificate.der
```

3.13.1.2.6 Certificate authority (CA)

Client-side certificates should be verified by a CA before use. The certificates can be verified before installing using the `openssl` tool.

3.13.1.2.7 Example of verification

In this verification example, a self-signed CA is used. The CA is stored in the file `ca.pem`.

```
$ openssl verify -CAfile ca.pem client_1024@example.com.pem
client_1024@example.com.pem: OK
```

For more options and deeper information about verification, see the `openssl` manual.

The u-connectXpress software supports certificates of the format PEM and DER. If the certificates are in another format, they must be converted before downloading. This can be done using the `openssl` application, see http://www.openssl.org for more information about this.

3.13.2 Transport Layer Security (TLS)

3.13.2.1 What is TLS

Transport Layer Security (TLS) – and its predecessor, Secure Sockets Layer (SSL), which is now deprecated by the Internet Engineering Task Force (IETF) – are cryptographic protocols that provide communications security over a computer network.

TLS 1.2 was defined in RFC 5246 in August 2008.

3.13.2.2 TLS handshake

When the connection starts, the record encapsulates a "control" protocol -the handshake messaging protocol. As shown in Figure 26, this handshaking protocol is used to exchange all the information required by both sides for the exchange of the actual application data by TLS. It defines the format of messages and the order of their exchange. These may vary according to the demands of the client and server, i.e., there are several possible procedures to set up the connection. This initial exchange results in a successful TLS connection (both parties ready to transfer application data with TLS.

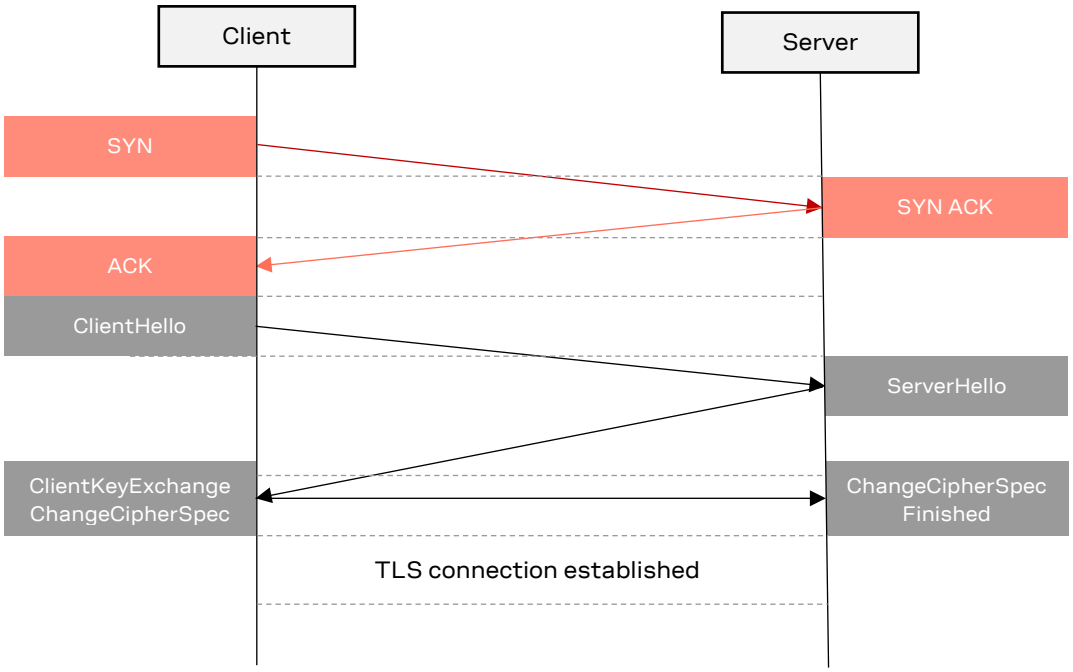


Figure 26: TLS handshake between client and server

3.13.3 Bluetooth security

There are several different security modes that support all kinds of use cases related to Bluetooth pairing procedures. For more information, see Bluetooth security application note [9].

☞ To mitigate certain vulnerabilities, the Bluetooth SIG recommends that product developers introduce language to user interfaces and/or documentation that warns users to not enter the numeric comparison value on the remote pairing device or to not enter the numeric comparison value anywhere.

3.13.4 IoT security

u-connectXpress supports the most commonly used security-modes:

- TLS 1-way handshake
- TLS 2-way handshake
- Certificate/key schemes
- User/password schemes
- SAS Tokens

For more information about these security modes, see the u-connectXpress IoT Cloud connectivity application note [20] and u-connectXpress MQTT application note [19].

3.14 Wireless Multidrop

With Wireless Multidrop, any local module can communicate with several devices simultaneously—without any need to install any additional software on the host system. Wireless Multidrop is automatically enabled when the module is in data mode.

All data sent over the UART interface from the local host to the module is wirelessly distributed to all connected remote devices. The data received from connected remote devices is subsequently forwarded to the host over UART interface. The data received from a connected remote device is not distributed to the other connected remote devices.

You can use Wireless Multidrop in the following scenarios:

- **Same to all**: The same data is sent from the Central device to all remote devices.
- **Poll one, retrieve data from one device:** The same data is sent from the Central device to all remote devices. Assuming the host implements a protocol, which allows addressable recipients, the application on the remote devices ensures that only the addressed device responds. For example, the Modbus serial communications protocol might use multidrop to connect a host to a Remote Terminal Unit (RTU).
- **Command one:** The same data is sent from the central device to all remote devices. Assuming the host implements a protocol, which allows addressable recipients, the applications on the remote devices ensure that only the addressed device takes action. An example of a higher-level protocol that can be used to accomplish this is Modbus RTU.

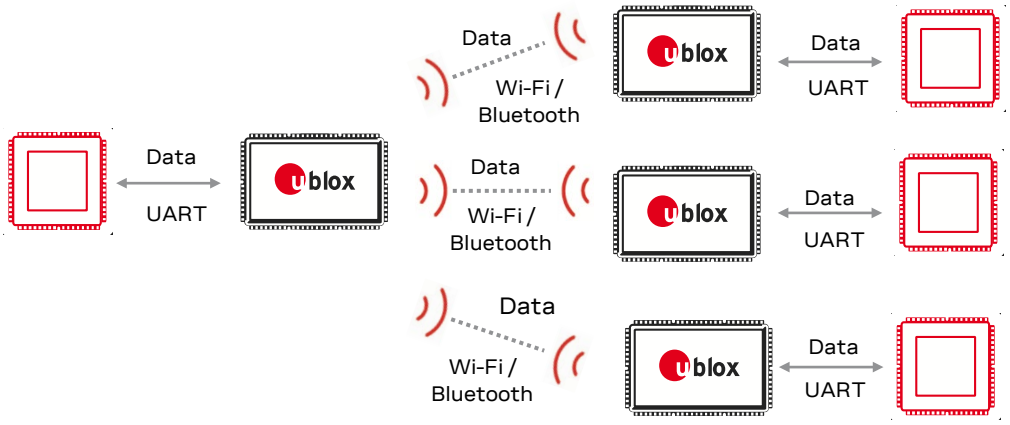


Figure 27: Wireless multidrop scenarios

4 Use cases

This chapter describes how u-blox short range stand-alone modules can be used in different use cases.

The examples include configuration details that as a precondition assume that the module has been set in Factory default mode `AT+UFACTORY`, if nothing else is stated. In some configurations, the default values for some parameters are assumed.

4.1 Wi-Fi connectivity

The u-blox short range stand-alone module enables connectivity to an existing wireless network acting as a Wi-Fi station.

4.1.1 Use case 1: Serial to Wi-Fi station

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

Let the module act as a Wi-Fi station to connect to the local area network (LAN). This can be useful to replace cables (serial connections) to improve working conditions and reduce costs in manufacturing industries.

For example, the module is placed in a tool used to mount bolts in cars. The host that is connected to the u-blox module starts identifying the bolt and sends information to the network or server using access points (AP). The server or network then returns the torque curve information and correct tool configuration to adjust the bolt. When done, the tool sends the logging file to the server.

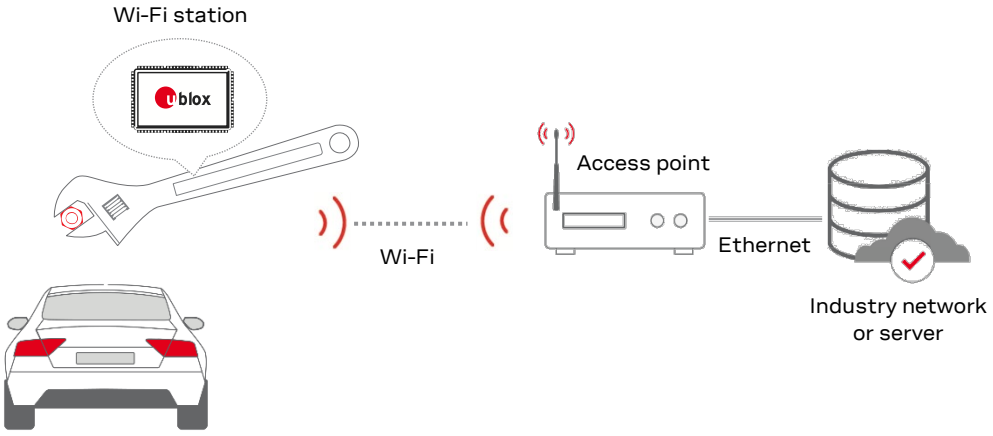


Figure 28: Example of u-connectXpress software as a Wi-Fi station to provide network connectivity

This use case is similar to the one described in Use case 1: Wi-Fi local area network enabler. The difference is that the u-blox short range stand-alone module acts as a Wi-Fi station to connect to an existing wireless network in this use case, while in the other acts as an AP to provide access to the network.

4.1.1.1 Configuration (not stored in the module)

	Instructions	AT command
1	Set SSID for the Network	`AT+UWSC=0,2,"myssid"`
2	Use WPA2 as authentication type	`AT+UWSC=0,5,2`
3	Use Password “mypassword”	`AT+UWSC=0,8,"mypassword"`
4	Enable DHCP client	`AT+UWSC=0,100,2`
5	Activate Wi-Fi Station configuration	`AT+UWSCA=0,3`
6	Wait for Wi-Fi interface to connect	`+UUWLE:0,112233445566,11`
7	Connect using TCP port 8080 on AP	`AT+UDCP="tcp://192.168.2.1:8080"`
8	Enter Data Mode to send data	`ATO1`
9	These settings are not stored in the flash memory of the module. The host needs to write this every time that the u-blox short range stand-alone module reboots	

4.1.1.2 Configuration (stored in the module)

Use the following configuration to make the u-blox short range stand-alone module store the configuration in the flash memory and automatically connect to a Wi-Fi network at power on.

	Instructions	AT command
1	Set Wi-Fi to be active at startup	`AT+UWSC=0,0,1`
2	Set SSID for the Network	`AT+UWSC=0,2,"myssid"`
3	Use WPA2 as authentication type	`AT+UWSC=0,5,2`
4	Use Password “mypassword”	`AT+UWSC=0,8,"mypassword"`
5	Enable DHCP client	`AT+UWSC=0,100,2`
6	Store the Wi-Fi Station configuration	`AT+UWSCA=0,1`
7	Set default remote peer to use TCP port 8080 on AP, using always connected. Use *optional* parameter ac-to to set timeout before reconnect attempt	`AT+UDDRP=0,"tcp://192.168.2.1:8080/?ac-to=5000",2`
8	Set startup mode to data mode	`AT+UMSM=1`
9	Store configuration to the startup database	`AT&W`
10	Reboot the u-blox short range module	`AT+CPWROFF`
	u-blox short range module restarts	
11	The settings are now stored in the flash memory of the module. On power up, the u-blox short range module connects to the network and the TCP connection whenever the module reboots. All data sent to the UART on the module is then sent to the remote IP address 192.168.2.1	

4.1.2 Use case 2: Serial to Wi-Fi access point

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

This use case is similar to Use case 1: Serial to Wi-Fi station. The difference here is that the u-blox short range stand-alone module acts as an AP to provide access to the network. In the other instance, the module acts as a Wi-Fi station to connect to an existing wireless network.

4.1.2.1 Configuration (not stored in the module)

	Instructions	AT command
1	Set SSID for the Network	`AT+UWAPC=0,2,"myssid"`
2	Set Channel 1 for the Network	`AT+UWAPC=0,4,1`
3	Set WPA2 Security for the Network	`AT+UWAPC=0,5,2,2`
4	Use Password “mypassword"	`AT+UWAPC=0,8,"mypassword"`
5	Enabled DHCP server	`AT+UWAPC=0,106,1`
6	Set server configuration id 1, using TCP and port 8080	`AT+UDSC=1,1,8080`
7	Activate Access Point configuration	`AT+UWAPCA=0,3`
8	Enter Data Mode to send data	`ATO1`
9	The u-blox short range module starts the Access Point and devices can now connect to the network with the SSID "UBXWifi". When Wi-Fi is connected and the network is up, the TCP listener on the module starts. The data sent to the TCP connection and transferred to the serial interface on the module.	

4.1.2.2 Configuration (stored in the module)

Use the following configuration to make the u-blox short range module store the configuration in the flash memory and automatically start a Wi-Fi network at power on.

	Instructions	AT command
1	Set Wi-Fi Access Point to be active at startup	`AT+UWAPC=0,0,1`
1	Set SSID for the Network	`AT+UWAPC=0,2,"myssid"`
2	Set Channel 1 for the Network	`AT+UWAPC=0,4,1`
3	Set WPA2 Security for the Network	`AT+UWAPC=0,5,2,2`
4	Use Password “mypassword"	`AT+UWAPC=0,8,"mypassword"`
5	Enabled DHCP server	`AT+UWAPC=0,106,1`
6	Store the Wi-Fi Access Point configuration	`AT+UWAPCA=0,1`
7	Set server configuration id 1, using TCP and port 8080	`AT+UDSC=1,1,8080`
8	Set startup mode to data mode	`AT+UMSM=1`
9	Store configuration to the startup database	`AT&W`
10	Reboot the u-blox short range module	`AT+CPWROFF`
11	The settings are now stored in the flash memory of the module. On power up, the u-blox short range module starts the Access Point and devices can now connect to the network with the SSID "myssid". When Wi-Fi is connected and the Network is up, the TCP listener on the module starts. The data sent to the TCP connection is transferred to the serial interface on the module.	

4.1.3 Use case 3: Serial to Wi-Fi (serial cable replacement)

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

By combining the Use case 1: Serial to Wi-Fi station with Use case 2: Serial to Wi-Fi access point, it is possible to make a serial cable replacement using Wi-Fi.

4.1.4 Use case 4: Serial PPP to Wi-Fi station

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

Use u-blox short range stand-alone modules to download or upload larger files using Wi-Fi instead of your cellular data plan. Let the u-blox short range module act as a Wi-Fi station to connect to a known Wi-Fi network when available.

Serial PPP is a protocol commonly used between a device and a cellular modem to provide Internet connectivity over UART. Since PPP is supported by u-blox short range stand-alone modules, it is easy to integrate the Wi-Fi connectivity using PPP to the Wi-Fi module.

For example, when a truck is within Wi-Fi range returning to the garage, the log files are uploaded and new driver instructions are received without any interaction from the driver.

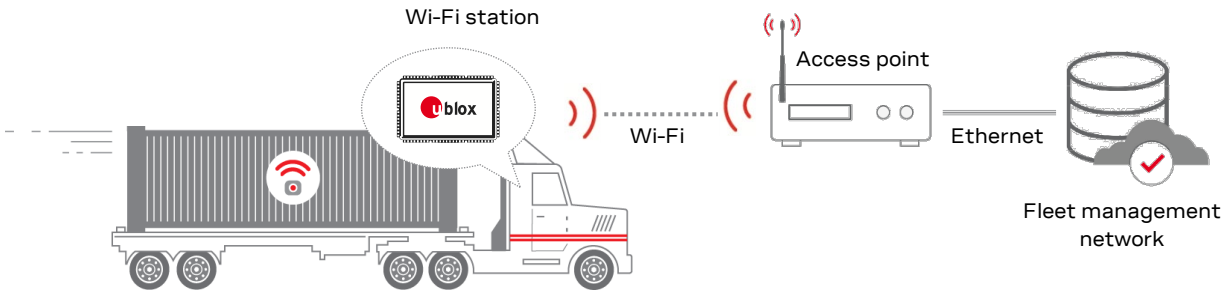


Figure 29: Example of a u-blox short range module acting as a Wi-Fi station to provide network connectivity

4.1.4.1 Configuration

	Instructions	AT command
1	Set PPP Network IP address as seen by the host for the PPP client.	`AT+UPPPC=101,172.30.0.252`
2	Set PPP Network Subnet mask for the client.	`AT+UPPPC=102,255.255.255.0`
3	For ODIN-W2 only: Optionally, disable DHCP relaying	`AT+UPPPC=107,1`
4	Activate the PPP configuration	`AT+UPPPCA=1`
5	Enter PPP Mode.	`ATO3`
6	Make sure the Serial Port in your host's software is closed.	
7	For Windows hosts, install the ODIN-W2 Windows Dial-up Modem Driver. This is only needed once.	
8	Connect the Dial-up modem that supports PPP client, such as the Windows built-in PPP client or Linux `pppd`. On Linux, you do this by killing `pppd`, and then restarting it with: `sudo pppd <port> 115200 crtscts noauth defaultroute usepeerdns` On Windows, you do this by creating a PPP modem with a dummy phone number and no username/password using the ODIN-W2 Windows Dial-up modem driver.	
	The module has now received the address 172.30.0.251 for the PPP network interface on the host, as described in the documentation for the `AT+UPPPC` command, and it listens on AT-commands from the host on UDP port 23. The address obtained may easily be verified using `ipconfig` on Windows, or `ifconfig` on Linux.	

	Instructions	AT command
	For testing the interface, make sure the host is not connected to any other network, and then ping to 172.30.0.252 and 172.30.0.251 from the host. Ping replies are expected.	
	To send and receive AT commands, transmit UDP packets from the host to 172.30.0.251 on port 23. For testing purposes, Netcat can be used. Transmit packets by entering the AT commands directly to the stdin of Netcat when started with the following parameters: `nc -u -C -p 47311 172.30.0.251 23` -u indicates that UDP packets are to be used, instead of TCP and -C that each packet shall be terminated by CRLF. -p is needed to ensure all responses are to be received on the same port of the host. For more information, see https://en.wikipedia.org/wiki/Netcat	
9	Send AT-command from the host to UDP port 23 to set the SSID for the Network	`AT+UWSC=0,2,"myssid"`
10	Optionally, set the desired password for WPA2 authentication (through UDP port 23).	`AT+UWSC=0,5,2` `AT+UWSC=0,8,"mypassword"`
11	Optionally, ensure the module always starts the Wi-Fi Station on module startup (through UDP port 23)	`AT+UWSC=0,0,1`
12	Store and activate the Wi-Fi Station configuration (through UDP port 23)	`AT+UWSCA=0,1` `AT+UWSCA=0,3`
13	When the module has connected to the AP, `+UUWLE:` and `+UUNU:` are reported. To verify network access and DNS from the host, ping a well-known service, such as `www.u-blox.com`	

14	Optionally, ensure the module always starts in PPP mode (through UDP port 23)	`AT+UMSM=3`
		`AT&W0`
		`ATI9`

4.1.5 Use case 5: RMII/Ethernet to Wi-Fi station bridge

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

The u-blox short range stand-alone modules can act as a Wi-Fi station to connect to an AP or home router to replace the Ethernet cable without any modification to the device.

When replacing an Ethernet cable, it is possible to use low-level RMII signaling without the need of an external PHY component to reduce costs. If access to the low-level RMII is not available, external PHY can easily be integrated and configured by the u-blox short range module.

In the home meter monitoring example shown in Figure 30, the home meter offers connectivity to a home meter monitoring server using a connected Ethernet cable to a switch or router. Then, you can easily monitor the device using an app on a smart device (mobile phone or tablet).

Figure 30: Example of a u-blox short range module acting as a Wi-Fi station to provide network connectivity

4.1.5.1 Configuration

This example configures the Wi-Fi Bridge in the u-blox short range module to route all Layer 2 traffic between the Wi-Fi station interface and the Ethernet interface. See u-connectXpress AT commands manual [6] for more information about the `AT+UBRGC` command and the parameters.

Due to the limitations of Wi-Fi, it is required to use the host Ethernet MAC address on the Wi-Fi interface. Restart the module after changing the MAC address.

☞ In this setup, it is not possible to access the u-blox short range module over the network interfaces; you can use only the UART interface.

	Instructions	AT command

1	Change the MAC address for the Wi-Fi interface. Example using "`ipconfig -all`" on a PC to check MAC address (112233AABBCC)
	`C:\>ipconfig -all`
	`Ethernet adapter Local Area Connection:`
	`Description . . . . . : Intel(R) Ethernet Connection I218-LM`
	`Physical Address. . . . . : 11-22-33-AA-BB-CC`
	`DHCP Enabled. . . . . : Yes`
	`Autoconfiguration Enabled . . . . : Yes`
		`AT+UMLA=2,112233AABBCC`

2	Store the configuration to startup database	`AT&W`
3	Reboot the module	`AT+CPWROFF`
4	Enable the Wi-Fi bridge between 1: Wi-Fi station and 3: Ethernet interface	`AT+UBRGC=0,1,1,3`
5	On NINA-W13 and NINA-W15, set a dummy static IP-address for the network bridge	`AT+UBRGC=0,100,1` `AT+UBRGC=0,101,192.168.43.15`
6	Optionally, store the configuration to flash and active on startup	`AT+UBRGC=0,0,1` `AT+UBRGCA=0,1`
7	Activate the bridge configuration: Use PHY (use `AT+UETHC=1,0` for RMII)	`AT+UBRGCA=0,3` `AT+UETHC=1,1`
8	If a PC is used as a host, you might have to disable Auto-negotiation on the PC using the `AT+UETHC=4,0` or use a switch between the Ethernet interface and the PC	`AT+UETHC=4,0`
9	Optionally, store the configuration to flash and active on startup	`AT+UETHC=0,1` `AT+UETHCA=1`
10	Activate the Ethernet configuration. Default values (100 Mbit, Full duplex, and Auto negotiation) are used in this example.	`AT+UETHCA=3`
11	Connect the Ethernet cable and wait for the interface to go up	`+UUETHLU`

	Instructions	AT command
12	Use an open network to configure the Wi-Fi Station	AT+UWSC=0,2,"myssid " AT+UWSC=0,5,1
13	Optionally, store the configuration to flash and active on startup	AT+UWSC=0,0,1 AT+UWSCA=0,1
14	Activate the Wi-Fi configuration	AT+UWSCA=0,3
15	Wait for the Wi-Fi interface to be connected	+UUWLE:0,112233445566,11

4.1.6 Use case 6: UDP connectivity

Supported modules	Software versions
ODIN-W260/W262	All
ODIN-W263	All
NINA-W13	All
NINA-W15	All

A UDP server can work in two modes:

- No connect mode: The server can only receive data and cannot return any data. Moreover, the sender of the incoming data cannot be identified. This mode is normally used with the UDP server in data mode. If several peers connect to the UDP server, the sent data is mixed over the UART – with no possibility to identify which peer that sent the data.
- Auto connect mode: In this mode, it is possible to identify the peer that sends the incoming data and also reply to the connecting peer – if the UDP server runs in Extended Data Mode. If the UDP server is in data mode the incoming data is mixed over the UART, any data sent from the UDP server is broadcast to all connected peers.

Before setting up the No connect mode example and Auto connect mode example, follow the procedures described in Use case 1: Serial to Wi-Fi station and Use case 2: Serial to Wi-Fi access point to set up one device as a Wi-Fi station and the other device as a Wi-Fi access point.

4.1.6.1 No connect mode example

	Instructions	AT command
1	**Device A**: Disable active server	AT+UDSC=1,0
2	**Device A**: Set up UDP server in no connect mode	AT+UDSC=1,2,5000,0
3	**Device B**: Connect from UDP Client	AT+UDCP=udp://192.168.2.1:5000/
4	**Device B**: Receiving connect event	+UUDPC:1,2,1,0.0.0.0,0,192.168.2.1 ,5000
5	**Device B**: Enter data mode	ATO1
6	**Device B**: Send some data	qwerty
7	**Device A:** Receiving connect event (if not already in data mode)	+UUDPC:1,2,1,0.0.0.0,5000,255.255. 255.255,0 *(Note broadcast address)*
8	**Device A:** Enter data mode	ATO1

Any sent data from B from now on will be seen on the UART of device A.

☞ And data sent from Device A before Device B enters data mode will be discarded by Device B.

4.1.6.2 Auto connect mode example

	Instructions	AT command
1	**Device A**: Disable active server	AT+UDSC=1,0
2	**Device A**: Set up UDP server in auto connect mode	AT+UDSC=1,2,5001,1
3	**Device A**: Enter ED Mode	ATO2

	Instructions	AT command
4	**Device B**: Connect from UDP Client	`AT+UDCP=udp://192.168.2.1:5001/`
5	**Device B**: Receiving connect event	`+UUDPC:1,2,1,0.0.0.0,0,192.168.2.1,5001`
6	**Device B**: Enter data mode	`ATO1`
7	**Device B**: Send some data	`qwerty`
8	**Device A:** Receiving connect event (in EDM format)	`+UUDPC:6,2,1,0.0.0.0,5001,192.168.2.100,55645` *(Note peer address)*
9	**Device A:** Receiving data from peer	`qwerty`

Any data received on the UDP server will be possible to identify via the EDM channel in the data event, and it is also possible to send to the individual peers using the EDM channel.

☞ s-center has support for ED mode and this setup can be experimented with using s-center.

4.2 Wi-Fi network sharing / Wi-Fi access point

You can easily create your own wireless network to enable connections to a device or network. u-blox short range stand-alone modules enable secure network access with WPA2 support when acting as a Wi-Fi AP.

4.2.1 Use case 1: Wi-Fi local area network enabler

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

Let the module act as a Wi-Fi AP to provide access to the local area network (LAN). Up to ten Wi-Fi stations can be connected simultaneously to the same ODIN-W2 or NINA-W15 module, and four NINA-W13 modules can get access the LAN. A wireless solution allows improved and flexible working conditions compared to a wired solution.

In the example given in Figure 31, the tool acts as a Wi-Fi station and is connected to the u-blox short range module that acts as an AP. The module then forwards the information from the tool to the network or server. The server or network then returns the torque curve information and correct tool configuration to adjust the bolt using the AP.

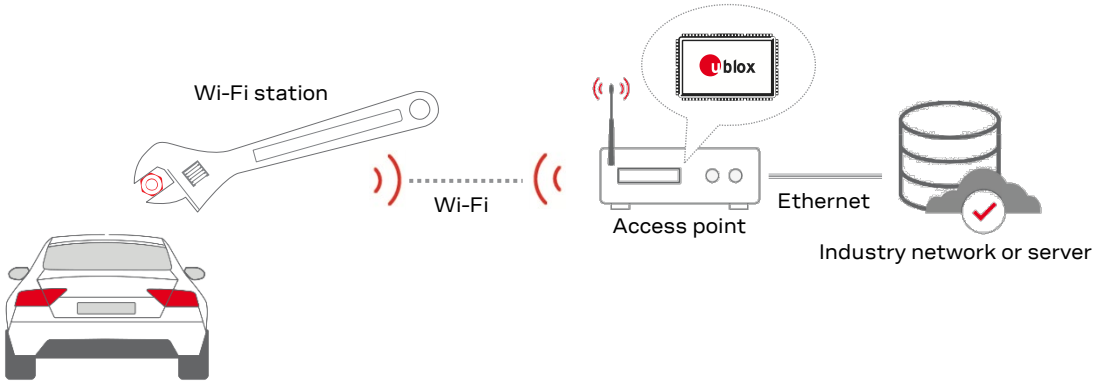


Figure 31: Example of a u-blox short range module acting as an access point to share network

☞ This use case is similar to Use case 1: Serial to Wi-Fi station. The difference is that the u-blox short range module acts as an AP to provide access to the network in this use case, while in the other, it acts as a Wi-Fi station to connect to an existing wireless network.

4.2.1.1 Configuration

	Instructions	AT command
1	Set Wi-Fi Access Point to be active at startup	`AT+UWAPC=0,0,1`
1	Set SSID for the Network	`AT+UWAPC=0,2,"myssid"`
2	Set Channel 1 for the Network	`AT+UWAPC=0,4,1`
3	Set WPA2 Security for the Network	`AT+UWAPC=0,5,2,2`
4	Use Password “`mypassword`"	`AT+UWAPC=0,8,"mypassword"`
5	Enable DHCP server	`AT+UWAPC=106,1`
6	Store Wi-Fi settings	`AT+UWAPCA=0,1`
7	Set server configuration id 1, using TCP and port 8080	`AT+UDSC=1,1,8080`
8	Set startup mode to data mode	`AT+UMSM=1`
9	Store configuration to startup database	`AT&W`
10	Reboot the u-blox short range module	`AT+CPWROFF`
11	The u-blox short range module starts the Access Point and devices can now connect to the network with the SSID "`UBXWifi`". When Wi-Fi is connected and the Network is up, the TCP listener on the module starts. The data sent to the TCP connection is transferred to the serial interface on module	

4.2.2 Use case 2: (Hosted) Wi-Fi tethering (hot spot)

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

Use the u-blox short range stand-alone modules together with a cellular modem to enable Internet access. Sharing the cellular network and allowing smart devices (mobile phone or tablet) to connect is also known as Wi-Fi tethering.

The example in Figure 32 shows a telematics box with a u-blox short range module connected to the cellular modem to enable Internet access to smart devices using Wi-Fi. The module acts as an AP and is connected to the application’s MCU interface using a RMII interface.

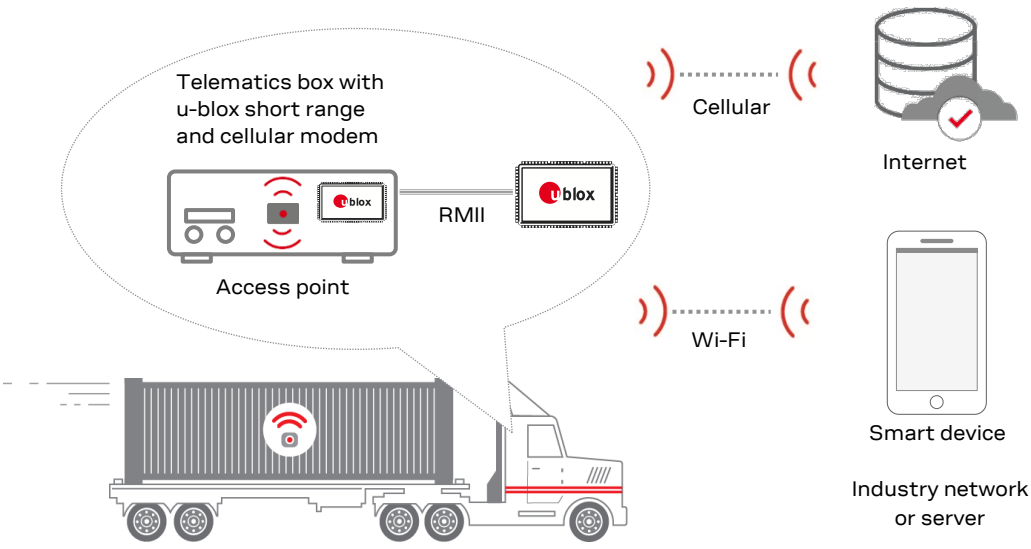


Figure 32: Example of a u-blox short range module acting as an access point to share network

4.2.2.1 Configuration

	Instructions	AT command
	Bridge configuration	
1	Enable bridging between 2: Wi-Fi Access Point and 3: Ethernet interface	`AT+UBRGC=0,1,2,3`
2	Active on startup (optional)	`AT+UBRGC=0,0,1`
3	Store configuration (optional)	`AT+UBRGCA=0,1`
4	Activate the bridge configuration	`AT+UBRGCA=0,3`
	Ethernet configuration	
5	Active on startup (optional)	`AT+UETHC=0,1`
7	Use RMII interface (Ethernet is default)	`AT+UETHC=1,0`
6	Store configuration (optional)	`AT+UETHCA=1`
8	Wait for the interface to go up	`+UUETHLU`
	Wi-Fi Access Point configuration	
9	Set SSID for the Network	`AT+UWAPC=0,2,"myssid "`
10	Set Channel 1 for the Network	`AT+UWAPC=0,4,1`
11	Set WPA2 Security for the Network	`AT+UWAPC=0,5,2,2`
12	Use Password “`mypassword`"	`AT+UWAPC=0,8,"mypassword"`
13	Enable DHCP server	`AT+UWAPC=106,1`
14	Active on startup	`AT+UWAPC=0,0,1`
15	Store configuration	`AT+UWAPCA=0,1`
16	Activate the Wi-Fi Station	`AT+UWAPCA=0,3`
17	Wait for the Wi-Fi Clients to connect	`+UUWLE:0,112233445566,11`

4.3 Wi-Fi and Bluetooth device configuration

u-blox short range stand-alone modules can be used to replace the Human Machine Interface (HMI) on all types of machines in all kinds of industries. This means that you can easily update the machine settings over a wireless connection from a laptop or smart device (mobile phone or tablet).

You are free to use the (iOS/Android) u-blox Bluetooth LE open-source app to discover, test and update local Bluetooth devices. And, as it is open source, you have the possibility to adapt the code to suit customer needs. Additionally, you can use the software to circumvent the need for any physical interface, like buttons or LCD displays, into the end-product design. Using this software, you have the good opportunity to increase both the efficiency and usability of your product offering.

4.3.1 Use case 1: Smartphone or tablet using Bluetooth Low Energy

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W15	All

To update remote-device settings with the u-blox Bluetooth LE app, you connect your mobile phone or tablet using Bluetooth Low Energy (LE) network technology. Bluetooth LE is battery effective and is supported by an abundance of smart devices.

u-blox short range modules offer wireless Internet access to remote devices over Wi-Fi while simultaneously accessing other devices over Bluetooth connections See also Configuration (stored in the module).

The example in Figure 33 shows how a u-blox short range module can be placed in a telematics box. If the configuration needs to be updated or modified in the box, like adding credentials for a new AP for instance, you can do so over a Bluetooth LE connection using the u-blox Bluetooth LE app.

Figure 33: Example of a u-blox short range module using wireless device configuration

4.3.1.1 Configuration

	Instructions	AT command
1	Enable Peripheral Mode	`AT+UBTLE=2`
2	Store	`AT&W`
3	Reboot to enable the Peripheral Mode	`AT+CPWROFF`
4	Change name to something easy to find	`AT+UBTLN="ODIN-W2 AP Setup"`
5	Enable AT Command over Air (COA) using Serial Port Service	`AT+UDSC=1,8,6`
6	Store and reboot	`AT&W`
7	Reboot to enable the Peripheral Mode	`AT+CPWROFF`
	Connect from another device that supports the u-blox Bluetooth Low Energy serial service SPS, such as a smartphone with the u-blox app. While sending AT commands, the CR (Carriage Return) must be included in the command.	
8	Now it is possible to provide AT command using the Serial Port Service on the u-blox short range module. An example of how to setup an access point on ODIN-W2 that allows clients to connect and locally communicate with each other is shown in the following instructions. Note: In NINA-W13 SW 2.0.0 and ODIN-W2 SW 7.0.0 or later, no bridge needs to be setup to let the connected clients communicate. In older ODIN-W2 versions, the bridge needs to be activated to allow communication between the connected clients.	
9	Using SPS configure the SSID	`AT+UWSC=0,2,"myssid"`
10	Configure to use WPA2	`AT+UWSC=0,5,2`
11	Configure Password	`AT+UWSC=0,8,"mypassword"`
12	Activate Wi-Fi Station	`AT+UWSCA=0,3`

4.3.2 Use case 2: Laptop using Wi-Fi

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

The u-blox short range stand-alone modules can act as an AP to provide wireless access to a machine for maintenance such as software updates or real-time diagnostics and control, from a laptop. This enables access to inaccessible machines, such as spreader or machines located in harsh environments using a Wi-Fi network. Using Wi-Fi improves the throughput if transferring a large amount of data.

As shown in Figure 34, the u-blox short range module is connected to the spreader and enables access from a laptop to control the spreader or to send new settings.

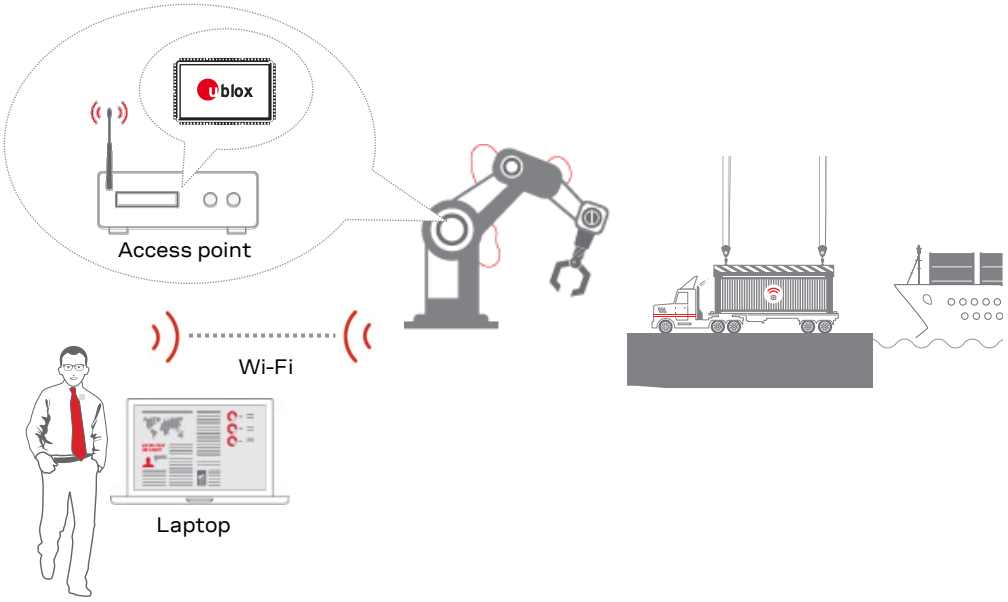


Figure 34: Example of a u-blox short range module using wireless device configuration

4.3.2.1 Configuration

	Instructions	AT command
1	Set SSID for the Network	`AT+UWAPC=0,2,"myssid"`
2	Set Channel 1 for the Network	`AT+UWAPC=0,4,1`
3	Set WPA2 Security for the Network	`AT+UWAPC=0,5,2,2`
4	Use Password “mypassword"	`AT+UWAPC=0,8,"mypassword"`
5	Active on startup	`AT+UWAPC=0,0,1`
6	Store the configuration	`AT+UWAPCA=0,1`
7	Activate the Wi-Fi Station	`AT+UWAPCA=0,3`
8	Wait for the Wi-Fi Access Point interface to be enabled. After this event has been received, the AP is ready and stations can connect	`+UUWAPU:0`
9	Set server configuration id 1, using TCP and port 8080	`AT+UDSC=1,1,8080`
10	Enter Data Mode to receive data on port 8080 from the remote device	`ATO1`

4.4 Bluetooth BR/EDR connectivity

4.4.1 Use case 1: Serial to Bluetooth

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-B2	All
NINA-W15	All

Establish a Bluetooth SPP connection between two u-blox short range stand-alone modules.

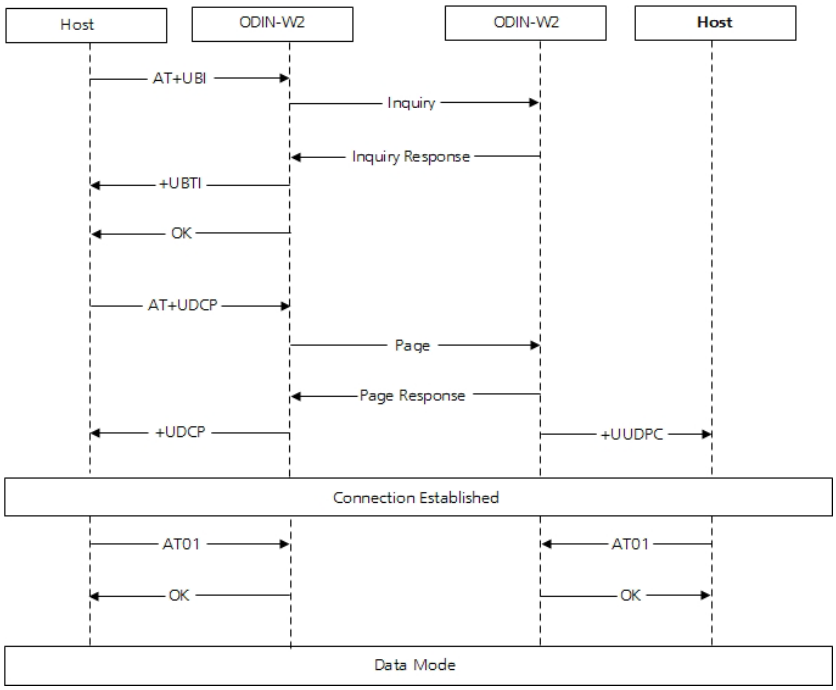


Figure 35: u-connectXpress software Bluetooth SPP connection

4.4.1.1 Configuration

	Instructions	AT commands
1	The u-blox module accepts incoming connection and replies on inquiry	
	Device 1	
2	Find the remote device using Inquiry	`AT+UBTI`
3	Find and write down the Bluetooth address for the remote device (device 2) to be used for the connection command	`+UBTI:222222222222,-52,000000,"Bluetooth Device"`
4	Device 1 connects via Bluetooth SPP with device 2. If no established connection, error code is shown	`AT+UDCP="spp://222222222222"` `+UDCP:1`
5	The connection event is received with information about connection type and Bluetooth address.	`+UUDPC:1,1,1,222222222222,669`
6	To enter data mode to be able to send and receive data	`ATO1`
	Device 2	
7	The connection event is received with information about connection type and Bluetooth address	`+UUDPC:1,1,1,111111111111,669`
8	To enter data mode to be able to send and receive data	`ATO1`

4.4.2 Use case 2: Serial to Bluetooth (serial cable replacement)

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-B2	All
NINA-W15	All

Establish a Bluetooth SPP connection to act as a serial cable replacement. It connects automatically and sends transparent data between two devices that stay connected, as shown in Figure 36.

Extend this example using two devices.

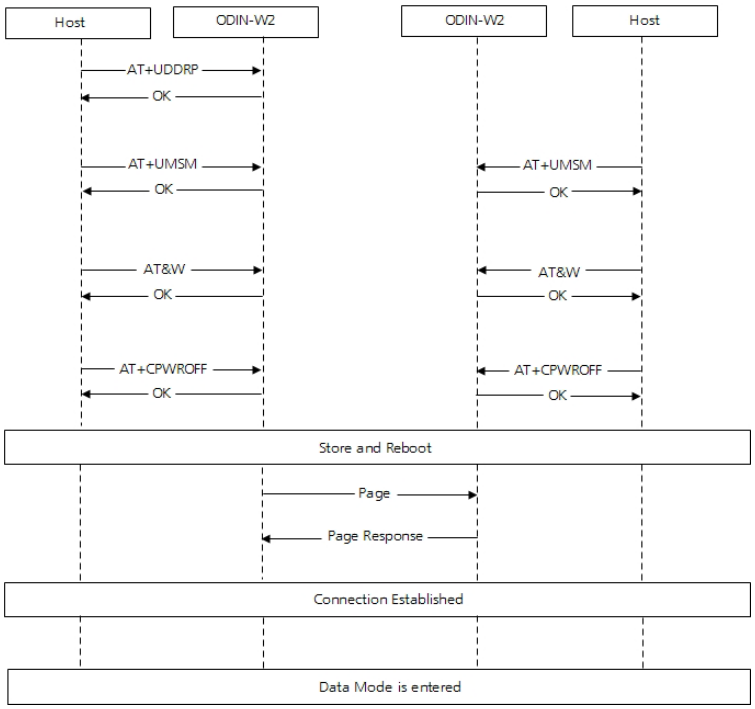


Figure 36: u-blox short range stand-alone modules Bluetooth SPP connection remote peer

4.4.2.1 Configuration

	Instructions	AT commands
1	Setup a default peer and configure with always connected parameter. To set the timeout before reconnect attempt, use the optional parameter `ac-to`	`AT+UDDRP=0,"spp://222222222222/?ac-to=5000",2`
2	Select startup mode and start in data mode	`AT+UMSM=1`
3	Store configuration in the startup database	`AT&W`
4	Reboot to use new settings	`AT+CPWROFF`

4.4.3 Use case 3: Bluetooth Personal Area Network (PAN user to smartphone)

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards

The Bluetooth BR/EDR Profile Personal Area Network (PAN) supports sending Ethernet data over Bluetooth (TCP or UDP). In this example, make sure that the Internet Connection Sharing or Personal Hotspot is enabled on the smartphone so that the smartphone can do a pairing. Then, open an application on the smartphone with a TCP listener on port 5003.

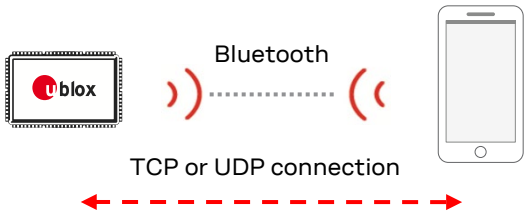


Figure 37: Bluetooth PAN connection to smartphone

4.4.3.1 Configuration

	Instruction to setup module (device 1) as PAN user	AT command
1	Set Central/Peripheral role to “let the connecting device select” (default value) to configure the smartphone as the Central	`AT+UBTMSP=1`
2	Set the local PAN role to PAN-PANU (client)	`AT+UBTPANC=1,0`
3	Set the remote PAN role to PAN-NAP (server)	`AT+UBTPANC=2,1`
4	Use the address of the smartphone	`AT+UBTPANC=4,112233445566p`
5	Use DHCP to retrieve an IP address	`AT+UBTPANC=100,2`
6	Activate the PAN client	`AT+UBTPANCA=3`
7	The phone may initiate a paring that the u-blox short range module accepts	`+UUBTB:48BF6B51D0C6,0`
8	The PAN Bluetooth connection is now created	`+UUBTPANLU:0,48BF6B51D0C6p`
9	The PAN network interface goes up to indicate that the network is up	`+UUNU:15`
10	Create a TCP connection to the smartphone, and make sure that the IP address is the one used by the phone. In this example the IP address of the phone is - `172.20.10.1`.	`AT+UDCP="tcp://172.20.10.1:5003"`
11	The TCP connection is connected to the application on the phone	`+UUDPC:1,2,0,172.20.10.2,49153, 172.20.10.1,5003`
12	Switch to the data mode to send data from the UART to the TCP connection	`ATO1`

4.4.4 Use case 4: Wi-Fi AP and Bluetooth PAN NAP Bridge

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards

It is possible to use both the Wi-Fi Access Point and PAN-NAP (Network Access Point) to bridge the two interfaces and use the DHCP server on the bridge to prove the IP address to both Bluetooth and Wi-Fi Station devices. For this use case, make sure your device supports the PAN PANU role. The Bluetooth PAN PANU device and the Wi-Fi Station can communicate using TCP or UDP protocols.

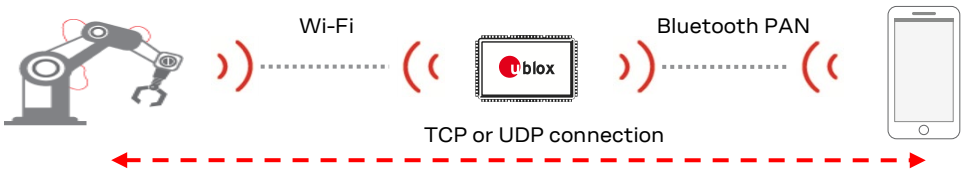


Figure 38: Bridge between PAN and Wi-Fi

4.4.4.1 Configuration

	Instruction to setup module	AT command
1	Bridge Wi-Fi Access Point and PAN interface	`AT+UBRGC=0,1,2,6`
2	Enable the Wi-Fi AP and PAN interfaces to accept the IP traffic	`AT+UBRGC=0,2,2,6`
3	Use static IP address on the Bridge	`AT+UBRGC=0,100,1`
4	Bridge Static IP address	`AT+UBRGC=0,101,192.168.0.50`
5	Bridge Network Mask	`AT+UBRGC=0,102,255.255.255.0`
6	Bridge Gateway	`AT+UBRGC=0,103,192.168.0.50`
7	Bridge primary DNS	`AT+UBRGC=0,104,192.168.0.50`
8	Bridge secondary DNS	`AT+UBRGC=0,105,0.0.0.0`
9	Enable DHCP Server on the Bridge	`AT+UBRGC=0,106,1`
10	Activate the Bridge network interface	`AT+UBRGCA=0,3`
11	Set role to Always Central, to get best performance (and to avoid scatternet)	`AT+UBTMSP=0`
12	Set the local PAN role to PAN-NAP (server)	`AT+UBTPANC=1,1`
13	Set the remote PAN role to PAN-PANU (client)	`AT+UBTPANC=2,0`
14	Activate the PAN-NAP (server) to allow incoming Bluetooth connections	`AT+UBTPANCA=3`
15	Wait for the PAN-PANU device to connect	`+UUBTPANLU:0,112233445566p`
16	Set network name SSID	`AT+UWAPC=0,2,"myssid"`
17	Use channel 6	`AT+UWAPC=0,4,6`
18	Use WPA2 for security	`AT+UWAPC=0,5,2,2`
19	Set the passkey	`AT+UWAPC=0,8,"mypassword"`
20	Activate the Access Point on ODIN-W2	`AT+UWAPCA=0,3`
21	Wait for the Station to connect	`+UUWAPSTAC:0,48BF6B51D0DC`
	Set up a TCP listener on port 5003 of the Wi-Fi Station device; if another ODIN-W2 is used, send the following command to enable this	`AT+UDSC=1,1,5003,0`
	From the smartphone using PAN-PANU, use an app that supports TCP connections and connect to the remote device using the IP address received from the Wi-Fi Access Point	

4.5 Bluetooth Low Energy specific use cases

☞ In the use cases described in this chapter the value for characteristics handles may vary between different moules and between SW versions.

☞ The ANNA-B1, ANNA-B4, NINA-B1, NINA-B3 and NINA-B4 modules have Bluetooth low energy peripheral enabled by default, which means this step can be skipped for these modules in some use cases.

4.5.1 Use case 1: Set up a GATT server / client

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards (**GATT client part only**)
ODIN-W263	v7.1.0 onwards (**GATT client part only**)
ANNA-B112	All
ANNA-B412	All
NINA-B1	v2.0.0 onwards
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All

See the u-connectXpress AT commands manual [6] for detailed description of the command parameters.

☞ When connecting to a GATT server from a mobile phone, it is occasionally necessary to refresh the Bluetooth cache in the phone to see the changes made.

The following example shows how to set up a GATT Server on one module (device A) and receive notifications of updated values at a GATT Client on a second (device B) module.

4.5.1.1 Set up a GATT server with a predefined service on device A

	Instructions	AT command
1	Enable Peripheral Role	`AT+UBTLE=2`
2	Store configuration and restart	`AT&W` `AT+CPWROFF`
3	Define a Heart Rate service	`AT+UBTGSER=180D` `+UBTGSER:30`
4	Add a Heart Rate measurement characteristic with notification support	`AT+UBTGCHA=2A37,10,1,1` `+UBTGCHA:32,33`

4.5.1.2 Use GATT client on device B to receive heart rate measurement values

	Instructions	AT command
1	Enable Central Role	`AT+UBTLE=1`
2	Store configuration and restart	`AT&W` `AT+CPWROFF`

3	Find the other device
	Identify the device in the list of shown devices
		`AT+UBTD=4,1`
		`+UBTD:112233445566p,-59,`
		`"NINA-B1 GATT Server"...`

4	Create an ACL connection	`AT+UBTACLC=112233445566` `+UUBTACLC:0,0,112233445566p`

5	Use the connection handle 0 from the +UUBTACLC and Discover Services	`AT+UBTGDP=0`
		`+UBTGDP:0,1,9,1800`
		`+UBTGDP:0,10,13,1801`
		`+UBTGDP:0,14,22,180A`
		`+UBTGDP:0,23,29,01D7E9014FF344`
		`E7838FE226B9E15624`
		`+UBTGDP:0,30,65535,180D`
		`OK`

	Instructions	AT command
6	After finding the “Heart Rate service” (180 D), use the start handle 30 and end handle 65535 from the +UBTGDP, and Discover all characteristics of service	AT+UBTGDCS=0,30,65535 +UBTGDCS:0,31,10,32,2A37
7	After finding the “Heart Rate Measurement characteristics” (2A37), use the value handle 32 from the +UBTGDCS and end handle 65535 from the +UBTGDP, and Discover all characteristic descriptors	AT+UBTGDCD=0,32,65535 +UBTGDCD:0,32,33,2902
8	After finding the “Client Characteristic Configuration descriptor” (2902), use the descriptor handle 33 from the +UBTGDCD, and subscribe to notifications of Heart Rate measurement value change	AT+UBTGWC=0,33,1

4.5.1.3 Update and notifying a new value of heart rate measurement

	Instructions	AT command
1	We can now set a new value (50) and send a notification from Device A (the GATT Server)	AT+UBTGSN=0,32,50
2	On Device B, an event is received when the remote side (Device A) sends a notification	+UUBTGN:0,32,50

4.5.2 Use case 2: Define GATT characteristics with user defined size

Supported modules	Software versions
ANNA-B112	v2.0.0 onwards
ANNA-B412	All
NINA-B1	v5.0.0 onwards
NINA-B31	v2.0.0 onwards
NINA-B41	All

This example shows how you can generate a GATT table with user defined characteristics of limited size. This can be used to save space and fit more characteristics in. The default size of a GATT characteristic if not explicitly specified is 244 bytes. The UUIDs used are examples generated by an online UUID generator. The user defined services and characteristics should have 128-bit UUIDs.

	Instructions	AT command
1	Enable Peripheral Role	AT+UBTLE=2
2	Store configuration and restart	AT&W AT+CPWROFF
3	Define GATT service	AT+UBTGSER=4906276bda6a4a6cbf9473c61b96433c
4	Define characteristics limited to one byte in size	AT+UBTGCHA=49af5250f17646c5b99aa163a672c042,12,1,1,00,1,1 AT+UBTGCHA=f8c7dee6fafe4a6785b0a09ba675815d,12,1,1,00,1,1 AT+UBTGCHA=85abed6630fe4b36aa3d32173dc69876,12,1,1,00,1,1 AT+UBTGCHA=8ff6916bddc041febf6c67b9c61fe33d,12,1,1,00,1,1 AT+UBTGCHA=388a90641a5a4759a714be213b69167a,12,1,1,00,1,1 AT+UBTGCHA=dc0dac92a3d3497ca1f5c98907f5f910,12,1,1,00,1,1 AT+UBTGCHA=3f8d918aa74f4beaa5dba1c4e0afb68d,12,1,1,00,1,1 AT+UBTGCHA=9db36411f2704258869da875c5f12c89,12,1,1,00,1,1 AT+UBTGCHA=55133e4c4fda48c59edd820ce62f94fd,12,1,1,00,1,1 AT+UBTGCHA=299c083da96f430f95064053cfe5ea2c,12,1,1,00,1,1 AT+UBTGCHA=b22a70b0a9cd4cba927e897e26b3b3e3,12,1,1,00,1,1 AT+UBTGCHA=d99577d0660d496a9716671c020b1cec,12,1,1,00,1,1 AT+UBTGCHA=d00984efbe2c436d8f66d578f5ce031a,12,1,1,00,1,1 AT+UBTGCHA=f4392b64acb64aac94eb84605083bf8b,12,1,1,00,1,1 AT+UBTGCHA=2e23a8824a554d8e8677196f7029a2c4,12,1,1,00,1,1 AT+UBTGCHA=846931bdc9084b268901a8523571b1d6,12,1,1,00,1,1 AT+UBTGCHA=26fad88089894444a92ad53b00e2393d,12,1,1,00,1,1 AT+UBTGCHA=c30e586d7aa84d0dac7e19b05cdbbb04,12,1,1,00,1,1 AT+UBTGCHA=bcb96978ed30450b993fa6123b039348,12,1,1,00,1,1 AT+UBTGCHA=914a15df56d84c0898d394b73565d26c,12,1,1,00,1,1 AT+UBTGCHA=e9404fdcac0f44f9b7cfa1dfa76687ae,12,1,1,00,1,1

Instructions	AT command
	AT+UBTGCHA=437afc6977fc4710a96a9b2f99656ad5,12,1,1,00,1,1 AT+UBTGCHA=ebe33ab3457e48279f7c6515050a0ea4,12,1,1,00,1,1 AT+UBTGCHA=e193377ef677435fb5f86c2be411e417,12,1,1,00,1,1 AT+UBTGCHA=7fd569e9701e4924bf2a4523bd65c4df,12,1,1,00,1,1 AT+UBTGCHA=a79d9d0c59cb437c9e6bd38231613f2b,12,1,1,00,1,1

☞ The number of characteristics that can be defined is limited by available RAM and depends on several factors. For example, what roles the device has, how many concurrent Bluetooth LE connections that are allowed, and so on.

Conditions	Value	Number of possible GATT elements
Role	Central + Peripheral	1 service + 29 characteristics
# of concurrent Bluetooth LE links	7	
Characteristics size	Default	
SPS enabled	No	

Table 12: Example of number of characteristics in NINA-B1 SW 5.0.x

4.5.3 Use case 3: Letting the system handle GATT characteristic values

Supported modules	Software versions
ANNA-B112	All
ANNA-B412	All
NINA-B1	v3.0.1 onwards
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All

This example describes the command instructions for configuring the system to automatically respond to GATT “reads” by allowing u-connectXpress handle the characteristic value.

	Instructions	AT command
1	**Device A:** Enable Peripheral Role	AT+UBTLE=2
2	**Device A:** Store configuration and restart	AT&W AT+CPWROFF
3	**Device A:** Define a Heart Rate service	AT+UBTGSER=180D +UBTGSER:30
4	**Device A:** Add a Heart Rate measurement characteristic with read/write properties, and an initial value (45).	AT+UBTGCHA=2A37,0A,1,1,45 +UBTGCHA:32,33
5	**Device B:** Configure as Central	AT+UBTLE=1
6	**Device B:** Connect to the GATT server on Device A	AT+UBTACLC=D4CA6EC596CA
7	**Device B:** Read the GATT characteristic	AT+UBTGR=0,32
8	**Device A+B:** Note that response is received on Device B without Device A having to send a response.	+UBTGR:0,32,45
9	**Device A:** Update the characteristic value	AT+UBTGSV=32,60
10	**Device B:** Read again	AT+UBTGR=0,32
11	**Device B:** Receive value	+UBTGR:0,32,60
12	**Device B:** Write an updated characteristic value to Device A.	AT+UBTGW=0,32,55
13	**Device A:** Receive an event notifying about the characteristic value update. Note that the value as read by clients is updated automatically.	+UUBTGRW:0,23,55,1

4.5.4 Use case 4: Long GATT writes

Supported modules	Software versions
ANNA-B112	All
ANNA-B412	All
NINA-B1	V2.0.0 onwards
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All
ODIN-W2	v3.0 onwards

A long write can be used when a GATT characteristic value that is longer than the available MTU size -3, which is the normal max size for a GATT write which needs to be written.

	Instructions	AT command
1	**Device A:** Enable Peripheral Role	`AT+UBTLE=2`
2	**Device A:** Store configuration and restart	`AT&W` `AT+CPWROFF`
3	**Device A:** Define a Heart Rate service	`AT+UBTGSER=180D` `+UBTGSER:30`
4	**Device A:** Add a Heart Rate measurement characteristic with read/write properties	`AT+UBTGCHA=2A37,0A,1,1` `+UBTGCHA:32,33`
5	**Device B:** Configure as Central	`AT+UBTLE=1`
6	**Device B:** Connect to the GATT server on Device A	`AT+UBTACLC=D4CA6EC596CA`

7	**Device B:** Send 2 long writes of 18 bytes each (MTU size is 23 by default, max size for a long write is MTU size -5).
	Note the offset and Final data indication on the second write.
		`AT+UBTGWL=0,32,123456789012345`
		`6789012345678901234567890123456,0,1,0`
		`AT+UBTGWL=0,32,789012345678901`
		`2345678901234561234567890,0,0,18`

8	**Device A:** Long write received. The ending options tag 2 indicates long write.	`+UUBTGRW:0,23,1234567890123456`
		`7890123456789012345678901234567`
		`8901234567890123456123456,2`

4.5.5 Use case 5: Set up the modules as beacons

Supported modules	Software versions
ANNA-B112	All
ANNA-B412	All
NINA-B1	All
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All

4.5.5.1 Setting up a module as iBeacon

iBeacon is a Manufacturer-specific event and contains a 128-bit UUID such as D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C.

☞ The byte order for the AT command is reverse byte order. The UUID together with the Major (further specifies a specific iBeacon and use case, 2 bytes), Minor (allows further subdivision of region or use case, 2 bytes) and the Tx power (1 byte) make up the iBeacon advertise packet.

Apple recommend having the Local Name in the Scan Response packet. For more information about iBeacon, see also reference [17].

☞ This is only an example use case. Apple requires an iBeacon license agreement if used in a product.

	Instructions	AT command
1	Enable Peripheral Role	`AT+UBTLE=2`
2	Store configuration and restart	`AT&W` `AT+CPWROFF`
3	Change advertise packet to include iBeacon	`AT+UBTAD=1AFF4C000215EBEFD08370A247C89837E7B5634DF52400010001C5`
4	Set empty Scan Response Data	`AT+UBTSD=00`

Scan for the iBeacon from a suitable application on your smartphone.

☞ Step 2 above is only valid for:

- ANNA-B1, NINA-B2, NINA-B3, NINA-W15 SW 3.0.0, or later
- NINA-B1 SW 6.0.0, or later

4.5.5.2 Setting up a module as an Eddystone beacon

☞ This example shows how to send an URL inside the advertisement packet. By nature, the size of the advertisement packet and the information given is limited.

The payload often contains a short version of a URL such `https://goo.gl/Aq18zF`. The link is encoded to save space, but most smartphones can use this information to navigate to the advertised URL. For more information about Eddystone, see reference [17].

	Instructions	AT command
1	Enable Peripheral Role	`AT+UBTLE=2`
2	Store configuration and restart	`AT&W` `AT+CPWROFF`
3	Change advertise packet to include Eddystone encoded URL (https://www.u-blox.com)	`AT+UBTAD=0303AAFE0D16AAFE10F801752D626C6F7807`
4	Change Local Name to “Bluetooth Device”	`AT+UBTSD=1109426C7565746F6F746820446576696365`

Scan for the Eddystone beacon from a suitable application on your smartphone.

4.5.6 Use case 6: Set up a module as a beacon with extended advertising

Supported modules	Software versions
ANNA-B112	v2.0.0
ANNA-B412	All
NINA-B1	v5.0.0
NINA-B31	All
NINA-B41	All

☞ This example shows how to send an URL inside the advertisement packet; by nature, the advertisement packet is limited in size and the information is also limited. With extended advertising we can assign longer advertising packets, up to 252 bytes.

	Instructions	AT command
1	Enable Extended advertising. This command requires a store and restart.	AT+UBTLECFG=29,1 AT&W AT+CPWROFF
2	Change advertise packet to include Eddystone encoded URL (https://www.u-blox.com/en/product/nina-b3-series)	AT+UBTAD=0303AAFE2616AAFE10F801752D626C6F7800656E2F70726F647563742F6E696E612D62332D73657269657300
3	Change Local Name to “Bluetooth Device”	AT+UBTSD=1109426C7565746F6F746820446576696365

Scan for the Eddystone beacon from a suitable application on your smartphone.

4.5.7 Use case 7: Connect two modules using 2 Mbit/s PHY

Supported modules	Software versions
ANNA-B112	All
ANNA-B412	All
NINA-B1	v4.0.0 onwards
NINA-B31	All
NINA-B41	All

Set up a link between two modules and change the link to 2 Mbit/s.

On both devices set preferred PHY:

	Instructions	AT command
1	Set 1 Mbps and 2 Mbps as preferred PHYs for Tx and Rx	AT+UBTLECFG=27,3 AT+UBTLECFG=28,3

Connecting from device A to B:

	Instructions	AT command
1	**Device A:** Set to Central role	AT+UBTLE=1
2	**Device A:** Store and reset	AT&W AT+CPWROFF
3	**Device A:** Connect to device B	AT+UDCP=sps://112233445566 +UDCP:1 OK +UUBTACLC:0,0,DBEF35897A91r +UUDPC:1,1,4,DBEF35897A91r,20
4	**Device A:** Request 2Mbps PHY for connection handle 0 A confirmation event is communicated if request is successful.	AT+UBTLEPHYR=0,2,2 +UUBTLEPHYU:0,0,2,2

4.5.8 Use case 8: Connect two modules and automatically switch to 2 Mbit/s PHY

Supported modules	Software versions
ANNA-B112	v2.0.0 onwards
ANNA-B412	All
NINA-B1	v5.0.0 onwards
NINA-B31	v2.0.0 onwards
NINA-B41	All

Set up a link between two modules and automatically change the link to 2 Mbit/s.

On both devices set preferred PHY:

	Instructions	AT command
1	Set 1 Mbps and 2 Mbps as preferred PHYs for Tx and Rx	`AT+UBTLECFG=27,3` `AT+UBTLECFG=28,3`
2	**Peripheral only:** Enable 1Mbps advertising with 2 Mbps secondary	`AT+UBTLECFG=29,3`
3	Store and reset	`AT&W` `AT+CPWROFF`

Connecting from device A to B:

	Instructions	AT command
1	**Device A:** Set to Central role	`AT+UBTLE=1`
2	**Device A:** Store and reset	`AT&W` `AT+CPWROFF`

3	**Device A:** Connect to device B	`AT+UDCP=sps://112233445566`
		`+UDCP:1`
		`OK`
		`+UUBTACLC:0,0,DBEF35897A91r`
		`+UUDPC:1,1,4,DBEF35897A91r,20`

4	Note that you get a PHY update event on both devices	`+UUBTLEPHYU:0,0,2,2`

4.5.9 Use case 9: Connect two modules using Coded PHY

Supported modules	Software versions
ANNA-B412	All
NINA-B31	All
NINA-B41	All

Set up a link between two modules and change the link to Coded PHY. Coded PHY is a feature that allows longer range by introducing Forward Error Correction (FEC) to the transmission.

Set preferred scan and advertising PHY on both devices:

	Instructions	AT command
1	Set Coded PHY as preferred PHY for Tx and Rx	`AT+UBTLECFG=27,4` `AT+UBTLECFG=28,4`
2	Set devices to advertise and scan on long rang (CODED) PHY	`AT+UBTLECFG=29,2`
3	Store and reset	`AT&W` `AT+CPWROFF`

Connecting from device A to B:

	Instructions	AT command
1	**Device A:** Set to Central role	`AT+UBTLE=1`
2	**Device A:** Store and reset	`AT&W` `AT+CPWROFF`

3	**Device A:** Connect to device B	`AT+UDCP=sps://112233445566`
		`+UDCP:1`
		`OK`
		`+UUBTACLC:0,0,DBEF35897A91r`
		`+UUDPC:1,1,4,DBEF35897A91r,20`

4.5.10 Use case 10: Change device information values

Supported modules	Software versions
ANNA-B112	v2.0.0 onwards
ANNA-B412	All
NINA-B1	v5.0.0 onwards
NINA-B31	v2.0.0 onwards
NINA-B41	All
NINA-W15	All

Change the characteristics values of the Device Information service (UUID 0x180A) to the values of your choice.

	Instructions	AT command
1	Read the current values of the Device Information service	AT+UBTLEDIS=1 AT+UBTLSEDIS=2 .. AT+UBTLEDIS=8
2	Set your specific Device Information values Manufacturer name Model number FW Revision SW Revision Serial Number System ID HW Revision PnP ID	 AT+UBTLEDIS=1,"u-blox" AT+UBTLEDIS=2,"NINA-B3" AT+UBTLEDIS=3,"5.0.0" AT+UBTLEDIS=4,"5.0.0" AT+UBTLEDIS=5,"1234" AT+UBTLEDIS=6,"1234567812345678" AT+UBTLEDIS=7,"HW1.A" AT+UBTLEDIS=8,"12345671234567"
3	Store and reset	AT&W AT+CPWROFF

Access your device from a smartphone using the u-blox Bluetooth LE app (or any other Bluetooth LE scanner application) and inspect the Device Information values.

☞ Earlier SW versions can also change Device Information but a limited set of values.

4.5.11 Use case 11: Bond two devices using passkey

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
ANNA-B112	All
ANNA-B412	All
NINA-B1	v2.0.0 onwards
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All

Two devices are bonded using passkey.

	Instruction to setup the module	AT command
1	**Device A:** Set Security mode 5 (keyboard only)	AT+UBTSM=5
2	**Device A:** Set as Central	AT+UBTLE=1
3	**Device A:** Store and restart	AT&W AT+CPWROFF

	Instruction to setup the module	AT command
4	**Device B:** Set Security mode 3 (display only)	AT+UBTSM=3
5	**Device B:** Set as Peripheral	AT+UBTLE=2
6	**Device B:** Store and restart	AT&W AT+CPWROFF
7	**Device A:** Initiate bonding with device B	AT+UBTB=112233445566,1
8	**Device B:** Note the passkey display event	+UUBTACLC:0,0,223344556677 +UUBTUPD: 223344556677,**840081**
9	**Device A:** Send response event with passkey displayed on device B	AT+UBTUPE=D4CA6E7233B5,1,840081
10	**Device A+B:** Bonding event indicates successful bonding	+UUBTB:<remote address>,0

Bonding is now completed.

4.5.12 Use case 12: Bond two devices with low energy secure connections

Supported modules	Software versions
ANNA-B112	v3.0.0 onwards
ANNA-B412	All
NINA-B1	v6.0.0 onwards
NINA-B31	v3.0.0 onwards
NINA-B41	All
NINA-W15, NINA-B2	v3.0.0 onwards

Low energy secure connections add extra security to the bonding phase of the connection. In this example, two devices are bonded using the Numeric Comparison association model.

☞ For ANNA-B1 SW2.0.0 and NINA-B1 SW5.0.0 refer to Appendix B.1, Bond two devices with Low Energy secure connections (old version).

	Instruction to setup the module	AT command
1	**Device A+B:** Enable Secure Connections in FIPS mode. This makes the device deny bonding with any device not supporting low energy secure connections.	AT+UBTST=2
2	**Device A:** Set Security mode 4 (Display Yes/No)	AT+UBTSM=4
3	**Device A:** Set as Central	AT+UBTLE=1
4	**Device A:** Store and restart	AT&W AT+CPWROFF
5	**Device B:** Set Security mode 4 (Display Yes/No)	AT+UBTSM=4
6	**Device B:** Set as Peripheral	AT+UBTLE=2
7	**Device B:** Store and restart	AT&W AT+CPWROFF
8	**Device A:** Initiate bonding with device B	AT+UBTB=112233445566,1
9	**Device A+B:** Note the passkey display event	+UUBTACLC:0,0, <remote address>, +UUBTUC: <remote address>,<passkey>
10	**Device A+B:** Send response event indicating passkey displayed on the devices match.	AT+UBTUC=<remote address>,1
11	**Device A+B:** Bonding event indicates successful bonding	+UUBTB:<remote address>,0

Bonding is now completed.

If one of the devices does not support low energy secure connections (AT+UBTST=0) the bonding is denied.

4.5.13 Use case 13: Bond two devices with out of band security

Supported modules	Software versions
ANNA-B112	All
ANNA-B412	All
NINA-B1	v3.0.1 onwards
NINA-B31	All
NINA-B41	All

OOB Bonding is a way to add extra protection for the bonding sequence by exchanging a temporary key using an out of band method.

	Instruction to setup the module	AT command
1	**Device A:** Set up as Peripheral	`AT+UBTLE=2`
2	**Device A:** Set Security mode 6 (OOB)	`AT+UBTSM=6`
3	**Device A:** Store and restart	`AT&W` `AT+CPWROFF`
4	**Device A:** Generate a random OOB temporary key	`AT+UBTOTK=0`
5	**Device A:** Read the random OOB Temporary Key	`AT+UBTOTK?` `+UBTOTK:`**`C8355BF87FC03B7AD482D0FA6F83F67A`**
6	**Device B:** Set Central Mode	`AT+UBTLE=1`
7	**Device B:** Set OOB security mode	`AT+UBTSM=6`
8	**Device B:** Store and restart	`AT&W` `AT+CPWROFF`
9	**Device B:** Input the OOB Temporary Key generated in Device A	`AT+UBTOTK=1,`**`C8355BF87FC03B7AD482D0FA6F83F67A`**
10	**Device B:** Bond with Device A	`AT+UBTB=112233445566p,1`

Bonding is now completed.

4.5.14 Use case 14: Set up Peripheral to accept connections from multiple Central nodes

Supported modules	Software versions
ANNA-B112	v3.0.0 onwards
ANNA-B412	All
NINA-B1	v6.0.0 onwards
NINA-B31	v3.0.0 onwards
NINA-B41	All
NINA-W15	v3.0.0 onwards
NINA-B2	v3.0.0 onwards

It is possible for a module in Peripheral mode to accept incoming connections from several centrals.

	Instruction to setup the modules	AT command
1	**Device A:** Set up as concurrent Central and Peripheral	`AT+UBTLE=3`
2	**Device A:** Set 4 concurrent Bluetooth LE links	`AT+UBTCFG=2,4`
3	**Device A:** Reserve 3 links for Peripheral role (only necessary in combined role)	`AT+UBTCFG=14,3`
4	**Device A:** Store and restart	`AT&W` `AT+CPWROFF`

	Instruction to setup the modules	AT command
5	**Device A:** Define a GATT service and characteristic (Heart Rate Sensor)	`AT+UBTGSER=180D` `AT+UBTGCHA=2A37,1A,1,1`
6	**Device B:** Set Central Mode	`AT+UBTLE=1`
7	**Device B:** Store and restart	`AT&W` `AT+CPWROFF`
8	**Device C:** Set Central Mode	`AT+UBTLE=1`
9	**Device C:** Store and restart	`AT&W` `AT+CPWROFF`
10	**Device B:** Connect to Device A	`AT+UBTACLC=CCF95784D1D2p`
11	**Device C:** Connect to Device A	`AT+UBTACLC=CCF95784D1D2p`
12	**Device A:** Incoming connections	`+UUBTACLC:1,0,EAAFD20D9FAAr` `+UUBTACLC:2,0,416A85B46F0Br`
13	**Device B:** Read value of GATT Characteristic	`AT+UBTGR=0,32`
14	**Device A:** Answer to read on connection handle 1	`+UUBTGRR:1,32` `AT+UBTGRR=1,45`
15	**Device C:** Read value of GATT Characteristic	`AT+UBTGR=0,32`
16	**Device A:** Answer to read on connection handle 2	`+UUBTGRR:2,32` `AT+UBTGRR=2,45`

Step 3 above reserves several connections for the Peripheral role in combined Central + Peripheral role. If using Peripheral mode only (`AT+UBTLE=2`) this step is not necessary.

4.5.15 Use case 15: Serial to Bluetooth low energy

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
ANNA-B112	All
ANNA-B412	All
NINA-B1	v2.0.0 onwards
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All

Establish a Bluetooth Low Energy SPS connection.

4.5.15.1 Configuration

	Instruction to setup the first module (device 1) as Peripheral	AT command
1	**Device 1:** Enable the Peripheral Role.	`AT+UBTLE=2`
2	Store the configuration.	`AT&W`
3	Restart the device.	`AT+CPWROFF`
4	Set server configuration ID 1 to Serial Port Service. Not necessary on B1/B3/B4.	`AT+UDSC=1,6`
5	Start the device in data mode.	`AT+UMSM=1`
6	Store the configuration.	`AT&W`
7	Restart the device.	`AT+CPWROFF`

	Instruction to setup the second module (device 2) as Central	AT command
1	**Device 2:** Enable the Central Role.	`AT+UBTLE=1`
2	Store the configuration.	`AT&W`
3	Restart the device.	`AT+CPWROFF`
4	Connect [what] using Serial Port Service. Use the address of Device 2.	`AT+UDCP="sps://222222222222"`
5	Enter data mode.	`ATO1`

It is also possible to connect from the Peripheral device by enabling the Serial Port service `AT+UDSC=1,6` on the Central device and then using the address from device 1 `AT+UDCP="sps://Device1"`.

In this case, the Serial Port Service must be enabled on the Central for it to work.

4.5.16 Use case 16: Serial to Bluetooth Low Energy (serial cable replacement)

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
ANNA-B112	All
ANNA-B412	All
NINA-B1	v2.0.0 onwards
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All

Establish a Bluetooth Low Energy SPS connection from the Central and using always connected to act as a serial cable replacement. Use Inquiry to find the address of the second device.

4.5.16.1 Configuration

	Instruction to setup module (device 1) as Central	AT command
1	**Device 1:** Enable the Central Role.	`AT+UBTLE=1`
2	Store the configuration.	`AT&W`
3	Restart the device.	`AT+CPWROFF`
4	Default peer using Serial Port Service and always connected. Use the address of Device 2. Use optional parameter ac-to to set timeout before reconnect attempt.	`AT+UDDRP=1,"sps://222222222222p/?ac-to=5000",2`

	Instruction to setup module (device 1) as Central	AT command
5	Start the device in data mode.	AT+UMSM=1
6	Store the configuration.	AT&W
7	Restart the device.	AT+CPWROFF

	Instruction to setup second module (device 2) as Peripheral	AT command
1	**Device 2:** Enable the Peripheral Role.	AT+UBTLE=2
2	Store the configuration.	AT&W
3	Restart the device.	AT+CPWROFF
4	Set the server configuration ID 1 to Serial Port Service. Not necessary on ANNA-B1/ANNA-B41/NINA-B1/NINA-B31/NINA-B41.	AT+UDSC=1,6
5	Start the device in data mode.	AT+UMSM=1
6	Store the configuration.	AT&W
7	Restart the device.	AT+CPWROFF

4.5.17 Use case 17: Connect two modules and use automatic PHY adaptation

Supported modules	Software versions
ANNA-B412	All
NINA-B31	v3.0.0 onwards
NINA-B41	All

For NINA-B3 there is an automatic switch between CODED PHY and 1 Mbps or 2 Mbps PHY based on the link quality. In order to enable this automatic switching CODED PHY and at least one of 1 Mbps or 2 Mbps PHY must be set as preferred PHY.

Set preferred TX and RX PHY on both devices:

	Instructions	AT command
1	Set Coded PHY and 1Mbps PHY as preferred PHY for Tx and Rx	AT+UBTLECFG=27,5 AT+UBTLECFG=28,5

Connecting from device A to B:

	Instructions	AT command
1	**Device A:** Set to Central role	AT+UBTLE=1
2	**Device A:** Store and reset	AT&W AT+CPWROFF
3	**Device A:** Connect to device B	AT+UDCP=sps://112233445566 +UDCP:1 OK +UUBTACLC:0,0,DBEF35897A91r +UUDPC:1,1,4,DBEF35897A91r,20

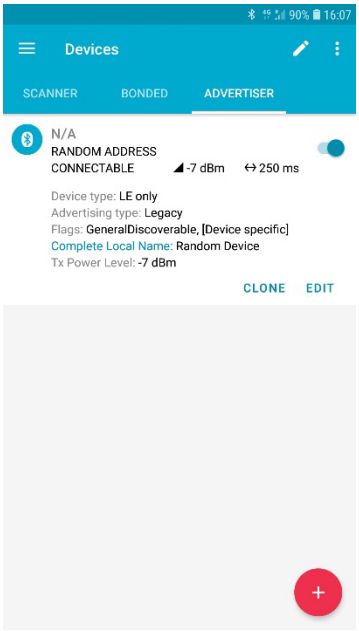
If link quality is deteriorating, the device now automatically moves over to CODED PHY. Similarly, the device moves from CODED PHY to 1 Mbps PHY when the link quality improves.

For details on the PHY switching algorithm refer to the u-connectXpress AT commands manual [6].

4.5.18 Use case 18: Connect to random resolvable address device using Identity Resolving Key (IRK)

Supported modules	Software versions
ANNA-B412	All
ANNA-B112	v4.0.0 onwards
NINA-B1	v7.0.0 onwards
NINA-B31	v4.0.0 onwards
NINA-B41	v2.0.0 onwards

During the bond process, keys are shared and exchanged between devices. When a device is configured to use Privacy and Random Resolvable Address, it periodically changes its address. Consequently, connecting to the device using the earlier address would not normally be possible. However, when the Identity Resolving Key (IRK) is available, the Central device can resolve the “new” random address to find the device which the IRK belongs and subsequently enable the connection.

	Instruction to setup the first device (device 1) as Peripheral	
1	Set one device as Peripheral with Random Address, for example using nRF Connect App [33]	

	Instruction to setup the second module (device 2) as Central	AT command
1	Device 2: Enable the Central Role.	`AT+UBTLE=1`
2	Store the configuration.	`AT&W`
3	Restart the device.	`AT+CPWROFF`

4	Find other devices, on this case “Random Device”	`AT+UBTD=1,1`
		`...`
		`+UBTD:72BA411A27D0r,-69,"Random Device",2,02011A0E0952616E646F6D20446576696365`
		`...`
		`OK`

5	Bond to the device.	`AT+UBTB=72BA411A27D0r,1`
		
		`+UUBTACLC:0,0,72BA411A27D0r`
		`+UUBTB:72BA411A27D0r,0`

	Wait a few minutes for the privacy to assign a new random address	

	Instruction to setup the second module (device 2) as Central	AT command
6	Confirm that IRK is available for the device 1. Note that <bd_addr> presents the public address rather than the random address.	AT+UBTBD=3 +UBTBD:B474433D059Cp,1,583E702939824B475FD878028DF3B5E2,1
7	Search for devices using the bonded devices filter.	AT+UBTD=5 +UBTD:B474433D059Cp,-47,"Random Device",2,02011A0E0952616E646F6D20446576696365 …
8	Connect using the public address.	AT+UBTACLC=B474433D059Cp OK +UUBTACLC:0,0,B474433D059Cp
9	Read the GATT services of Device 1.	AT+UBTGDP=0 +UBTGDP:0,1,3,1801 +UBTGDP:0,20,26,1800 +UBTGDP:0,40,65535,1805 OK

	Verify the Privacy and Random address (device 3)	AT command
1	Device 3: **S**can for Bluetooth LE devices. As the address of the random device changes frequently, this address is different from that it was previously.	AT+UBTD=1,1 ... +UBTD:65F18175C85Br,-63,"Random Device",2,02011A0E0952616E646F6D20446576696365 ... OK

4.6 IoT use cases

4.6.1 Use case 1: Connect using TLS connection

Supported modules	Software versions
ODIN-W260/W262	v7.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W15	All
NINA-W13	v2.1.0 onwards

Connections that use Transport Layer Security (TLS) provide a way to make sure the data that is sent and received is secure and encrypted. TLS is getting more common for different types of connections, like HTTPS, MQTT, and cloud services.

Here are some examples for different types of supported connections:

Connection	AT command
TCP	AT+UDCP=tcp://www.test.com:80
TLS Encryption (no validation)	AT+UDCP=tcp://www.test.com:443/?encr=1
TLS 1-way authentication:	AT+UDCP=tcp://www.test.com:443/?ca=ca_root.crt
TLS 2-way-authentication:	AT+UDCP=tcp://www.test.com:443/?ca=ca_root.crt&cert=client.pem&privKey=client.key

To upload the certificates, use the AT+USECMNG command. See also Certificate upload.

The following parameters can be used to configure and setup up the TLS connection:

- encr: TLS encryption without validating certificates if set to 1; for example, encr=1
- ca: Server CA for gateway to validate the server; for example, ca=ca.pem
- cert: Gateway client certificate; for example, cert=client.pem
- privKey: Gateway client private key; for example, privKey=client.key

It is possible to set the minimum level of TLS version to support using AT command. For example, to use TLS version 1.2 only, and not allow TLS 1.0 or TLS 1.1, set the query `encr=3`.

- Minimum TLS version 1.0 = 1
- Minimum TLS version 1.1 = 2
- Minimum TLS version 1.2 = 3

☞ Note that TLS version 1.3 is currently not supported.

For scenarios where `AT+UDHTTP` or `AT+UDHTTPE` is insufficient, it is possible to create a HTTPS connection directly in a TLS stream to a webserver.

	Instructions	AT command

1	Set the appropriate SSID and authentication for the network. In this case, WPA2 with password "`mypassword`".	`AT+UWSC=0,2,"myssid"`
		`AT+UWSC=0,5,2`
		`AT+UWSC=0,8,"mypassword"`

2	Activate Wi-Fi Station configuration.	`AT+UWSCA=0,3`
	Wait for Wi-Fi interface to connect	`+UUWLE:0,112233445566,11`
3	Connect to a HTTPS peer	`AT+UDCP="tcp://www.u-blox.com:443/?encr=1"`
	Wait for peer handle and peer connected event	`+UDCP:1` `+UUDPC:1,2,0,10.12.71.73,60060,52.218.236.201,443`
4	Enter Data Mode	`ATO1`

5	In Data Mode, send a HTTP request, including the linefeed separating the header and body of the request.	`GET /en HTTP/1.1`
		`Host: www.u-blox.com`
		`Connection: keep-alive`

	Receive response from the host	`HTTP/1.1 200 OK`
		`Date: Tue, 11 Feb 2020 12:07:51 GMT`
		`Content-Type: text/html; charset=utf-8`
		`Transfer-Encoding: chunked`
		`Connection: keep-alive`
		`Set-Cookie:`
		`__cfduid=db003981b55c0970ba631dd4772fa639f1581422871`
		`; expires=Thu, 12-Mar-20 12:07:51 GMT; path=/;`
		`domain=.u-blox.com; HttpOnly; SameSite=Lax`
		`X-Drupal-Cache: MISS`
		`Expires: Sun, 19 Nov 1978 05:00:00 GMT`
		`Cache-Control: max-age=3600, must-revalidate`
		`X-Content-Type-Options: nosniff`
		`Content-Language: en-US`
		`X-Frame-Options: SAMEORIGIN`
		`X-UA-Compatible: IE=edge`
		`Link: <https://www.u-blox.com/en>;`
		`rel="canonical",<https://www.u-blox.com/en>;`
		`rel="shortlink"`
		`Vary: Accept-Encoding`
		`X-Varnish: 402459529`
		`Age: 405`
		`Via: 1.1 varnish-v4`
		`X-Varnish-Cache: MISS`
		`X-AMAZEEIO: ublox1.compact frontend>varnish>backend`
		`Strict-Transport-Security: max-age=0`
		`CF-Cache-Status: HIT`
		`Expect-CT: max-age=604800, report-`
		`uri="https://report-uri.cloudflare.com/cdn-`
		`cgi/beacon/expect-ct"`
		`Server: cloudflare`
		`CF-RAY: 5636346fc9613d5f-CPH`
		
		`7c22`
		
		`<!DOCTYPE html>`
		`[and the rest of the page follows ... ]`

4.6.2 Use case 2: MQTT-SN gateway

Supported modules	Software versions
ODIN-W260/W262	v7.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W15	All

As shown in Figure 39, an MQTT-SN (SN=Sensor Network) gateway enables end devices with no TCP/TLS support to communicate with an MQTT broker (which requires TCP/TLS). The end device connects to the gateway using a serial connection, such as Bluetooth SPS or SPP, and communicates using the MQTT-SN protocol. The gateway then converts the MQTT-SN packets from the end-device to MQTT packets, to the Broker, and vice versa.

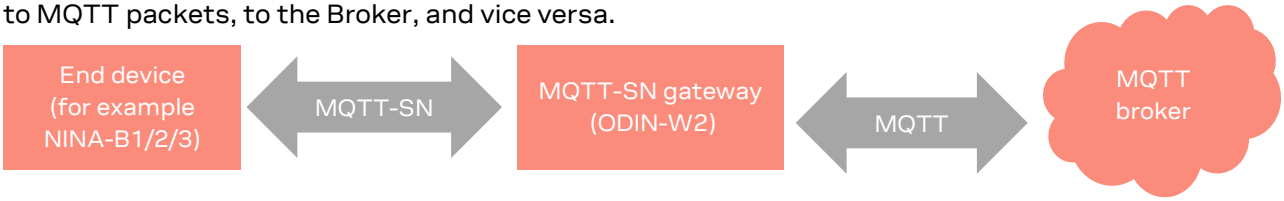


Figure 39: End device to MQTT broker communication

For information about using MQTT-SN, see also the u-connectXpress MQTT application note [19].

4.6.3 Use case 3: MQTT client gateway

Supported modules	Software versions
ODIN-W260/W262	v7.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W15	All
NINA-W13	v2.1.0 onwards

As shown in Figure 40, the gateway can also be configured as an MQTT client gateway rather than a full MQTT-SN gateway. In this configuration, it is possible for a host to transmit and receive MQTT data on the UART transparently.

In data mode, transmitted data is published to one configured publish topic and received data is received from one configured subscribe topic.

In EDM (extended data mode), it is possible to configure one published topic and one subscribed topic for every EDM channel. The host can then transmit and receive data on separate channels and consequently publish and subscribe to as many topics defined by the channels.

For NINA-W13 v3.0.0 and NINA-W15 v3.0.0 modules and later, it is possible to subscribe to all sub-topics in data mode. In this case, each received value is preceded with a topic-identifier.

Figure 40: Host to MQTT broker communication

More information about how to use MQTT Client Gateway is found in the u-connectXpress MQTT application note [19].

4.6.4 Use case 4: Connect to IBM Watson IoT platform

Supported modules	Software versions
ODIN-W260/W262	v7.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W15	All
NINA-W13	v2.1.0 onwards

See also the u-connectXpress IoT Cloud connectivity application note [20].

4.6.5 Use case 5: Connect to Amazon AWS IoT core

Supported modules	Software versions
ODIN-W260/W262	v7.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W15	All
NINA-W13	v2.1.0 onwards

See also u-connectXpress IoT Cloud connectivity application note [20].

4.6.6 Use case 6: Connect to Microsoft Azure IoT hub

Supported modules	Software versions
ODIN-W260/W262	v7.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W15	All
NINA-W13	v2.1.0 onwards

See also u-connectXpress IoT Cloud connectivity application note [20].

4.6.7 Use case 7: HTTP/HTTPS client GET JSON data

This example shows HTTPS GET to obtain data from an HTTPS API peer using the module as a Wi-Fi Station.

Supported modules	Software versions
NINA-W13	v3.0.0 onwards
NINA-W15	v3.0.0 onwards

For more information about the interfaces, see also the u-connectXpress AT commands manual [6].

	Instructions	AT command

1	Set the appropriate SSID and authentication for the network. In this case, WPA2 with password “mypassword”.	`AT+UWSC=0,2,"myssid"`
		`AT+UWSC=0,5,2`
		`AT+UWSC=0,8,"mypassword"`

2	Activate Wi-Fi Station configuration.	`AT+UWSCA=0,3`
	Wait for Wi-Fi interface to connect	`+UUWLE:0,112233445566,11`

	Instructions	AT command
3	Create a remote peer that uses unauthenticated HTTPS over TCP when issuing HTTP/HTTPS requests.	`AT+UDCP="http-tcp://jsonplaceholder.typicode.com:443/?encr=1"`
	Wait for peer handle for the peer.	`+UUDPC:1,3,0,::,0,jsonplaceholder.typicode.com,443`
4	Issue a HTTPS GET request to the peer for the "`/todos/1`" API endpoint.	`AT+UDHTTP=1,0,"/todos/1"`

	Wait for the response from the peer.
	Here, the 200 OK response returns 83 bytes, using the default UTF-8 encoding.
		`+UUDHTTP:1,200,83,application/json; charset=utf-8,{`
		`"userId": 1,`
		`"id": 1,`
		`"title": "delectus aut autem",`
		`"completed": false`
		`}`

4.6.8 Use case 8: HTTP/HTTPS client POST JSON data

Supported modules	Software versions
NINA-W13	v3.0.0 onwards
NINA-W15	v3.0.0 onwards

☞ This example shows HTTPS POST to add data to an HTTPS API peer using the module as a Wi-Fi Station. For more information about the interfaces, see the u-connectXpress AT commands manual [6].

	Instructions	AT command

1	Set the appropriate SSID and authentication for the network. In this case, WPA2 with password "mypassword"	`AT+UWSC=0,2,"myssid"`
		`AT+UWSC=0,5,2`
		`AT+UWSC=0,8,"mypassword"`

2	Activate Wi-Fi Station configuration	`AT+UWSCA=0,3`
	Wait for Wi-Fi interface to connect	`+UUWLE:0,112233445566,11`
3	Configure a remote peer to use unauthenticated HTTPS over TCP when issuing HTTP/HTTPS requests	`AT+UDCP="http-tcp://jsonplaceholder.typicode.com:443/?encr=1"`
	Wait for peer handle for the peer	`+UUDPC:1,3,0,::,0,jsonplaceholder.typicode.com,443`
4	Issue a HTTPS POST request to the peer for the "`/todos`" API endpoint	`AT+UDHTTPE=1,1,"/todos","application/json; charset=utf-8",55`
	Wait for the response data prompt ">"	`>`
5	Send the specified number of bytes	`{"userId": 1,"title": "lorem ipsum","completed": false}`

	Wait for the confirmation response from the peer	`+UUDHTTP:1,201,78,application/json; charset=utf-8,{`
		`"userId": 1,`
		`"title": "lorem ipsum",`
		`"completed": false,`
		`"id": 201`
		`}`

4.6.9 Use case 9: System time using host clock

Supported modules	Software versions
NINA-W13	v3.0.0 onwards
NINA-W15	v3.0.0 onwards
NINA-B2	v3.0.0 onwards

☞ This example shows how to get system time after setting it from the host. For more information about the interfaces, see the u-connectXpress AT commands manual [6].

	Instructions	AT command
1	Get current system time in seconds	`AT+UMST=0`
	Response if requested approximate 1 minute after power on	`+UMST:"0000003C"`
2	Set the system time to 2020-07-14 15:35:58	`AT+UMSTS="5F0DD0DC"`
3	Get the current system time in seconds	`AT+UMST=0`
	Current system time	`+UMST:"5F0DD10A"`
4	Get the current system time in ISO 8601 format	`AT+UMST=1`
	Current system time	`+UMST:"2020-07-14T15:36:44"`

4.6.10 Use case 10: System time using NTP

Supported modules	Software versions
NINA-W13	v3.0.0 onwards
NINA-W15	v3.0.0 onwards

☞ This example shows how to get system time and keeping it up to date with NTP using the module as a Wi-Fi Station. For more information about the interfaces, see the u-connectXpress AT commands manual [6].

	Instructions	AT command

1	Set the appropriate SSID and authentication for the network. In this case, WPA2 with password “mypassword”, ensuring the module reconnects to the network upon startup.	`AT+UWSC=0,1,1`
		`AT+UWSC=0,2,"myssid"`
		`AT+UWSC=0,5,2`
		`AT+UWSC=0,8,"mypassword"`

2	Store and activate Wi-Fi Station configuration.	`AT+UWSCA=0,1` `AT+UWSCA=0,3`
	Wait for Wi-Fi interface to connect	`+UUWLE:0,112233445566,11`
3	Configure the NTP server to use	`AT+UNNTS=0,"pool.ntp.org"`
4	Enable the NTP client	`AT+UNNT=1,0`
5	Get the current time	`AT+UMST=1`
	Current time (GMT) is returned.	`+UMST:"2020-07-14T15:08:09"`
6	Store NTP client configuration, to ensure time is recalibrated after power loss.	`AT&W`

4.7 Other use cases

4.7.1 Use case 1: Ethernet to Wi-Fi access point bridge

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

☞ This example configures the bridge in a u-blox short range module to route all Layer 2 traffic between the Wi-Fi AP interface and an Ethernet interface. For more information about the interfaces, see the u-connectXpress AT commands manual [6].

☞ In this setup, it is not possible to access the u-blox short range module over the network interfaces; you can use only the UART interface.

4.7.1.1 Configuration

	Instructions	AT command
1	Enable bridge between 2: Wi-Fi AP and 3: Ethernet interface.	`AT+UBRGC=0,1,2,3`
2	Optionally, store the configuration to flash and active on startup.	`AT+UBRGC=0,0,1` `AT+UBRGCA=0,1`

3	Activate the bridge configuration using:
	PHY, or
	RMII
		`AT+UBRGCA=0,3`
		`AT+UETHC=1,1`
		`AT+UETHC=1,0`

4	Optionally, store the configuration to flash and active on startup.	`AT+UETHC=0,1` `AT+UETHCA=1`
5	Activate the Ethernet configuration. Default values (100 Mbit, Full duplex and Auto negotiation) are used in this example.	`AT+UETHCA=3`
6	If a PHY is used, connect the Ethernet cable and wait for the interface to start.	`+UUETHLU`

7	Configure the Wi-Fi AP. In this example, there is no security and set SSID to “myssid”. Note that the IP address is not to be used when the bridge is activated.	`AT+UWAPC=0,2,"myssid"`
		`AT+UWAPC=0,4,1`
		`AT+UWAPC=0,5,1,1`

8	Optionally, store the configuration to flash and active on startup.	`AT+UWAPC=0,0,1` `AT+UWAPCA=0,1`
9	Activate the Wi-Fi configuration.	`AT+UWAPCA=0,3`
10	Enable the Wi-Fi AP interface.	`+UUWAPU:0`
11	Connect a Wi-Fi station device such as a smartphone or another u-blox short range module configured as a Wi-Fi station. The device should now receive an IP address from the DHCP server from the network connected to the Ethernet interface.	`+UUWAPSTAC:0,D0A637C90E9E`

4.7.2 Use case 2: Wi-Fi access point to serial PPP

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

Serial PPP is a protocol commonly used between a device and a cellular modem to provide Internet connectivity over UART. Since PPP is supported by u-blox short range stand-alone modules, it is easy to integrate the Wi-Fi connectivity using PPP to the Wi-Fi module.

To provide an embedded webserver for end-user configuration of the host, u-blox short range stand-alone modules can provide Wi-Fi connectivity to any host capable of hosting its own IP-stack. For example, the module presents a Wi-Fi network with a pre-defined SSID and gateway IP number.

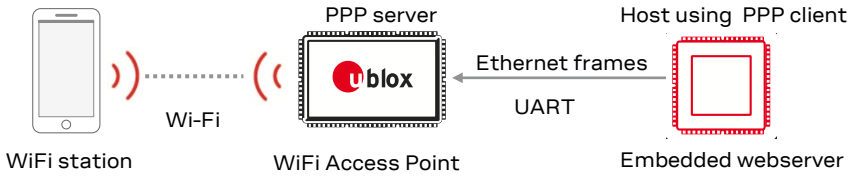


Figure 41: Example of a u-blox short range module acting as a Wi-Fi access point to provide network connectivity

4.7.2.1 Configuration

	Instructions	AT command
1	PPP Network IP address as seen by the host for the PPP client.	`AT+UPPPC=101,172.30.0.252`
2	PPP Network Subnet mask for the client.	`AT+UPPPC=102,255.255.255.0`
3	For ODIN-W2 only: Optionally, disable DHCP relaying	`AT+UPPPC=107,1`
4	Activate the PPP configuration	`AT+UPPPCA=1`
5	Set the SSID for the Network.	`AT+UWAPC=0,2,"myssid"`
6	Optionally, set the WiFi Channel	`AT+UWAPC=0,4,1`
7	Set the desired password for WPA2 authentication	`AT+UWAPC=0,8,"mypassword"` `AT+UWAPC=0,5,2,2`
8	Optionally, set the IP address to use as the station's gateway on the Wi-Fi address	`AT+UWAPC=0,101,192.168.2.1`
9	Optionally, set the subnet mask to be used on the Wi-Fi	`AT+UWAPC=0,102,255.255.255.0`
10	Enable the DHCP server	`AT+UWAPC=0,106,1`
11	Optionally, enable Address conflict detection	`AT+UWAPC=0,107,1`
12	Optionally, ensure the module always starts the Wi-Fi AP on module startup	`AT+UWAPC=0,0,1`
13	Store and activate Wi-Fi AP configuration	`AT+UWAPCA=0,1` `AT+UWAPCA=0,3`
	`+UUWAPU:0` and `+UUNU` is issued	
14	Enter the PPP Mode.	`ATO3`
15	Make sure the Serial Port in your host's software is closed.	
16	For Windows hosts, install the ODIN-W2 Windows Dial-up Modem Driver. This is only needed once.	
17	Connect the Dial-up Modem that supports PPP Client such as Windows built in PPP Client or Linux `pppd`. On Linux, this is done by killing `pppd`, then restarting it using `sudo pppd <port> 115200 crtscts noauth defaultroute usepeerdns` On Windows, this is instead done by creating a PPP modem with a dummy phone number and no username/password using the ODIN-W2 Windows Dial-up Modem Driver.	
	The module has now received the address 172.30.0.251 for the PPP network interface on the host, as described in the documentation for the `AT+UPPPC` command, and it listens on AT commands from the host on UDP port 23. The address obtained may easily be verified using `ipconfig` on Windows, or `ifconfig` on Linux. For testing the interface, make sure the host is not connected to any other network, and then send pings to 172.30.0.252 and 172.30.0.251 from the host. Ping replies are received.	
	To send and receive AT commands, transmit UDP packets from the host to 172.30.0.251 on port 23. For testing purposes, Netcat can be used. Transmit packets by typing the AT commands directly to the stdin of Netcat when started with the following parameters: `nc -u -C -p 47311 172.30.0.251 23` -u indicates that UDP packets are to be used, instead of TCP and -C that each packet shall be terminated by CRLF. -p is needed to ensure all responses are to be received on the same port of the host. See also https://en.wikipedia.org/wiki/Netcat	
18	Send a trivial AT-command from the host to UDP port 23 to test the connection	`ATI9`

	Instructions	AT command
19	Connect a station to the AP, using the defined password. `+UUWAPSTAC:` is issued over UDP when the station connects. The station receives an IP address based on the IP address specified above, such as 192.168.2.100. The subsequent gateway address is similar to one specified above, such as 192.168.2.1.	

	To verify, start a server on the host. You can use `netcat` if the host OS is Windows or Linux:
	`nc -C -l 12345`
	Connect a client on the station to the server on the host using the gateway address, as specified above. Again, you can use netcat on the station:
	`nc -C 192.168.2.1 12345`
	☞ On NINA-W13 and NINA-W15, it is the PPP client IP number that is exposed instead of the gateway address.
	Text typed to the `netcat` stdin on the station appears as output on the host console, and vice versa.	

20	Optionally, ensure the module always starts in PPP mode (via UDP port 23).	`AT+UMSM=3`
		`AT&W0`
		`AT+CPWROFF`

4.7.3 Use case 3: Ethernet to UART

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

4.7.3.1 Configuration

	Instructions	AT command
1	Use Static IP Address.	`AT+UETHC=100,1`
2	Use 192.168.0.101 as IP Address.	`AT+UETHC=101,192.168.0.101`
3	Use 255.255.0.0 as Subnet Mask.	`AT+UETHC=102,255.255.0.0`
4	Use 192.168.0.1 as Gateway.	`AT+UETHC=103,192.168.0.1`
5	Use Ethernet interface.	`AT+UETHC=1,1`
6	Activate the Ethernet settings.	`AT+UETHCA=3`
7	Enable AT Commands on TCP Port 23.	`AT+UDSC=1,1,23`

Use the IP address 192.168.0.100 on the PC.

The u-blox short range module now accepts the TCP connection on port 23, and all data is sent to the UART. Ensure that the carriage return "\r" is send together with the AT command like "AT\r".

- Example using `Netcat: nc -c 192.168.0.101 23`
- More information about Netcat: https://en.wikipedia.org/wiki/Netcat

4.7.4 Use case 4: Wi-Fi station via EAP-TLS to enterprise security

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
NINA-W13	v2.0.0 onwards
NINA-W15	All

To connect to enterprise security using EAP-TLS, which is considered a highly secure Wi-Fi connection, the u-blox short range module must upload a client certificate obtained by the network administrator. To prevent man-in-the-middle attacks, it is also recommended to upload the CA root certificate and validate this against the server certificate that was sent to the u-blox short range module during the connection setup – this is the default behavior. If the server certificate is not available or if this is not required (though less secure), it can be disabled by the `AT+UWSC=<id>,15,0` command. The server validation is also valid for PEAP enterprise security Wi-Fi connections.

4.7.4.1 Certificate upload

To upload the (CA) root certificate, use the `AT+USECMNG=0,0,<file_name>,<file_size>` command.

Command: The u-blox short range module responds with an ">" and then waits for the file to be sent in binary format.

Hardware flow control on the UART for high baud rates above 115200 is recommended. Hardware flow control is not necessary for UART baud rates of115200 or lower.

After the download has been completed (and all bytes in the `<file_size>` have been received), the u-blox short range module replies with a `+USECMNG` as shown in the example below. It also returns the MD5 hash of the file in DER-format. The host then verifies that the file has downloaded properly to the module.

☞ If the certificate is downloaded in PEM-format, which is also supported, the certificate must be converted to DER-format before the MD5 can be verified (on the host).

Example of MD5 hash (128 bit):

```
+USECMNG:0,0,"ca.der","621279af9b9b144acb61c3237be6fb82"
```

Example to upload the CA Root certificate (CA):

```
AT+USECMNG=0,0, "ca.der ",1024
```

Example to upload the client certificate (CC):

```
AT+USECMNG=0,1,client.der,2048
```

Example to upload the private key (PK):

```
AT+USECMNG=0,2,private_key.der,1024,"my_password"
```

Figure 42: Wi-Fi Station Enterprise security EAP-TLS

	Instruction to setup module	AT command
1	Use an open network to configure the Wi-Fi Station.	`AT+UWSC=0,2,"myssid"`
2	Set security to EAP-TLS	`AT+UWSC=0,5,5`
3	Select the Client Certificate that should be uploaded	`AT+UWSC=0,12,"client.der"`
4	Select the Private Key that should be uploaded	`AT+UWSC=0,13,"private_key.der"`
5	Select the CA Root Certificate to use in server validation	`AT+UWSC=0,14,"ca.der"`
6	Enable server certificate validation against CA root certificate	`AT+UWSC=0,15,1`
7	Activate the Wi-Fi configuration	`AT+UWSCA=0,3`

4.7.5 Use case 5: NFC links

Supported modules	Software versions
ANNA-B112	All
ANNA-B412	All
NINA-B1	v2.0.0 onwards
NINA-B31	All
NINA-B41	All

☞ For this use case, you must mount the NFC antenna on your evaluation kit (EVK).

4.7.5.1 NFC web link

Store a web link to the NFC tag; this web link is opened in the browser of your NFC enabled smartphone when you touch the NFC antenna of the module with your smartphone.

	Instruction to setup module	AT command
1	Set URI to NFC tag	`AT+UNFCURI=1,https://www.u-blox.com`
2	Enable NFC URI	`AT+UNFCEN=2`

Touch the NFC antenna with your NFC enabled smartphone to go to the u-blox website.

4.7.5.2 Launching an application with NFC

Store the name of an Android application package to the NFC tag; the application is opened on your NFC enabled smartphone when you touch the NFC antenna of the EVK with your smart phone.

The Android package name is the unique identifier of an application. It can be found by searching for the app and finding the Google Play web page for the app. For example, the web page for Google Calendar is: `https://play.google.com/store/apps/details?id=com.google.android.calendar`.

	Instruction to setup module	AT command
1	Set URI to NFC tag	`AT+UNFCURI=2,com.google.android.calendar`
2	Enable NFC URI	`AT+UNFCEN=2`

Touch the NFC antenna with your NFC enabled smartphone to open the Google calendar application.

4.7.6 Use case 6: Over the air configuration

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
ANNA-B112	All
ANNA-B412	All
NINA-B1	v2.0.0 onwards
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All

The modules can be set up to allow remote configuration. This enables a remote device to send AT commands over the air interface. In this example, the u-blox SPS interface is used for configuration.

Device to be configured over the air (Device A):

	Instruction to setup module	AT command
1	Set as Bluetooth LE Peripheral	`AT+UBTLE=2`
2	Store and restart	`AT&W` `AT+CPWROFF`
3	Enable SPS on server id 0 (**ODIN-W2, NINA-B2, NINA-W15**)	`AT+UDSC=0,0` `AT+UDSC=0,6`
4	Enable remote configuration on server id 0	`AT+UDSF=0,1`
5	Go to data mode	`ATO1`

Device B:

	Instruction to setup module	AT command
1	Set as Bluetooth LE Central	`AT+UBTLE=1`
2	Store and restart	`AT&W` `AT+CPWROFF`
3	Change escape character from default '+' (ASCII 43) to '-' (ASCII 45) to make the escape sequence pass through Device A without getting detected as an escape sequence at Device B	`ATS2=45`
4	If you do not know the address of Device A scan for it	`AT+UBTD=4,1`
5	Create an SPS connection to Device A.	`AT+UDCP=sps://<device A address>`
6	Go to data mode	`ATO1`

7	Enter configuration mode by sending escape sequence of Device A. The escape sequence needs to be sent within 200 ms, so copy it form editor and paste into terminal. You should get an OK response to indicate remote device in command mode	`1. 1s of silence`
		`2. +++`
		`3. 1s of silence`
		`OK`

8	Read the Bluetooth address of Device A	`AT+UMLA=1` `+UMLA:D4CA6E706EE2`

4.7.7 Use case 7: Read and write GPIO pins

Supported modules	Software versions
ODIN-W260/W262	v5.0.0 onwards
ODIN-W263	v7.1.0 onwards
ANNA-B112	All
ANNA-B412	All
NINA-B1	v2.0.0 onwards
NINA-B2	All
NINA-B31	All
NINA-B41	All
NINA-W15	All
NINA-W13	All

There are AT commands available to read and write the values of GPIO pins. For further information, see the respective system integration manual [4][5][11][12][13][14].

The easiest way to test this is to connect two GPIOs on your EVK and just use one. In this example, the GPIO numbers used are tried on a NINA-B112 EVK.

	Instruction to setup module	AT command
1	Check current configuration status of available GPIOs (result shows that all GPIOs are disabled)	AT+UGPIOC? +UGPIOC:2,255 +UGPIOC:3,255 +UGPIOC:4,255 +UGPIOC:5,255 +UGPIOC:24,255 +UGPIOC:25,255 +UGPIOC:27,255 OK
2	Set pin 2 as an input, no resistor activated	AT+UGPIOC=2,1,0
3	Read the current value of pin 2 current value is 0 (zero)	AT+UGPIOR=2
4	Set pin 3 as an output with initial value set to 0 (zero)	AT+UGPIOC=3,0,0
5	Set pin 3 to 1 (one)	AT+UGPIOW=3,1

4.7.8 Use case 8: Wi-Fi vendor-specific information element scanning

Supported modules	Software versions
NINA-W13	v2.0.0 onwards
NINA-W15	All

Vendor-specific Information Element(s) (IE) can be scanned. IEs are scanned using filters. Filters can match one or several IEs from the same or different access point(s). IEs are typically transmitted in other device beacons and probe responses.

Although scanning can be done independent of whether the module is configured as an AP or a station, these examples assume that the module is a station, and that two APs are configured as specified in Use case 9: Wi-Fi vendor-specific information element insertion. In this scenario, the access points broadcast to two different IEs that each use different SSIDs on different channels.

	Instructions	AT command
1	Scan for IEs belonging to vendor with OUI CC:F9:57 on all channels on all SSIDs.	AT+UWSCANIE="",CCF957
	Wait for scan result.	+UWSCANIE:D4CA6EC58C27,6,CCF95741752D626C6F78 +UWSCANIE:D4CA6EC58C27,6,CCF957424E494E412D573135 +UWSCANIE:D4CA6EFD9D5F,11,CCF95741752D636F6E6E656374587072657373 +UWSCANIE:D4CA6EFD9D5F,11,CCF95742752D636F6E6E656374536372697074 OK
2	Scan for IEs belonging to vendor with OUI CC:F9:57 on all channels on all SSIDs, but only with vendor-specific type 41 ('A').	AT+UWSCANIE="",CCF95741
	Wait for scan result.	+UWSCANIE:D4CA6EC58C27,6,CCF95741752D626C6F78 +UWSCANIE:D4CA6EFD9D5F,11,CCF95741752D636F6E6E656374587072657373 OK
3	Scan for IEs belonging to vendor with OUI CC:F9:57 on all channels on all SSIDs, but only with vendor-specific type 41 ('A') or 42 ('B'), and where the payload begins with 752D ('u-').	AT+UWSCANIE="",CCF95741752D,CCF95742752D
	Wait for scan result.	+UWSCANIE:D4CA6EC58C27,6,CCF95741752D626C6F78 +UWSCANIE:D4CA6EFD9D5F,11,CCF95741752D636F6E6E656374587072657373 +UWSCANIE:D4CA6EFD9D5F,11,CCF95742752D636F6E6E656374536372697074 OK

4.7.9 Use case 9: Wi-Fi vendor-specific information element insertion

Supported modules	Software versions
NINA-W13	v2.1.0 onwards
NINA-W15	v2.1.0 onwards

It is possible to define Vendor-specific Information Element(s) (IE) that may optionally be transmitted in the Wi-Fi beacon. In this example, two access points broadcast two different Information Elements ("IE") each, using different SSIDs on different channels.

The access points are configured as follows:

	Access point 1	Access point 2
SSID	"myssid"	"myssid2"
Channel	6	11
MAC-address	D4:CA:6E:C5:8C:27	D4:CA:6E:FD:9D:5F
OUI	CC:F9:57	CC:F9:57
IE 0, Vendor-specific Type	41 ('A')	41 ('A')
IE 0, Vendor- specific IE	752D626C6F78 ("u-blox")	752D636F6E6E656374587072657373 ("u-connectXpress")
IE 1, Vendor-specific Type	42 ('B')	42 ('B')
IE 1, Vendor- specific IE	4E494E412D573135 ("NINA-W15")	752D636F6E6E656374536372697074 ("u-connectXpress")

Configure the first access point as shown in the table above:

	Configure the Access Point 1	AT command
1	Set SSID for the Network.	AT+UWAPC=0,2,"myssid"
2	Optionally, store the configuration to flash and active on startup.	AT+UWSC=0,0,1 AT+UWSCA=0,1
3	Activate Wi-Fi AP configuration.	AT+UWAPCA=0,3
4	Wait for Wi-Fi interface to connect.	+UUWAPU:0 +UUNU:12
5	Configure IE 0 with Vendor-specific Type 41 ('A'), and Vendor-specific IE 752D626C6F78 ("u-blox")	AT+UWVSIE=0,1,CCF957,41,752D626C6F78
6	Configure IE 1 with Vendor-specific Type 42 ('B'), and Vendor-specific IE 4E494E412D573135 ("NINA-W15")	AT+UWVSIE=1,1,CCF957,42,4E494E412D573135
7	Optionally, store the IE configuration and restart	AT&W AT+CPWROFF

Configure the second access point as shown the table above:

	Configure the Access Point 2	AT command
1	Set SSID for the Network.	AT+UWAPC=0,2,"myssid2"
2	Optionally, store the configuration to flash and active on startup.	AT+UWSC=0,0,1 AT+UWSCA=0,1
3	Activate Wi-Fi AP configuration.	AT+UWAPCA=0,3
4	Wait for Wi-Fi interface to connect.	+UUWAPU:0 +UUNU:12
5	Configure IE 0 with Vendor-specific Type 41 ('A'), and Vendor-specific IE 752D636F6E6E656374587072657373 ("u-connectXpress")	AT+UWVSIE=0,1,CCF957,41,752D636F6E6E656374587072657373

	Configure the Access Point 2	AT command
6	Configure IE 1 with Vendor-specific Type 42 ('B'), and Vendor-specific IE 752D636F6E6E656374536372697074 ("u-connectXpress")	AT+UWVSIE=1,1,CCF957,42,752D636F6E6E656374536372697074
7	Optionally, store the IE configuration and restart	AT&W AT+CPWROFF

A station can now scan, filter, and detect the defined IEs described in Use case 8: Wi-Fi vendor-specific information element scanning.

	Instructions	AT command
1	Set SSID for the Network.	AT+UWSC=0,2,"myssid"
2	Activate Wi-Fi Station configuration	AT+UWSCA=0,3
3	Wait for Wi-Fi interface to connect	+UUWLE:0,D4CA6EC58C27,6
4	Scan for IEs belonging to vendor with OUI CC:F9:57 on all channels on all SSIDs	AT+UWSCANIE="", CCF957
	Wait for scan result.	+UWSCANIE:D4CA6EC58C27,6,CCF95741752D626C6F78 +UWSCANIE:D4CA6EC58C27,6,CCF957424E494E412D573135 +UWSCANIE:D4CA6EFD9D5F,11,CCF95741752D636F6E6E656374587072657373 +UWSCANIE:D4CA6EFD9D5F,11,CCF95742752D636F6E6E656374536372697074 OK

4.7.10 Use case 10: Bind an SPI stream over TCP

Supported modules	Software versions
NINA-W13	v3.0.0 onwards
NINA-W15	v3.0.0 onwards
NINA-B2 (not supporting Wi-Fi TCP stream used in this example)	v3.0.0 onwards

It is possible to connect a host to one of the supported modules using SPI. This can be done in different modes, as described in the application note [32]. Only SPI slave mode is currently supported in u-blox modules.

The recommended way to connect a u-blox module over SPI bus is by using the u-blox defined control protocol described in the application note [32].

To test this example on a u-blox EVK you need to connect an SPI master that can run the control protocol to the EVK, using patch cables that connect to the EVK pin headers. For information about the correct pins to use, refer to the appropriate data sheet [23] [27] [28] for your module.

To connect a host to one of the supported modules using SPI, you also need a Wi-Fi Access Point and TCP server, as used in the configuration example shown in Figure 8.

Use the following procedure to configure the AP and TCP server and connect the SPI slave to the Wi-Fi access point.

	Instructions	AT command
1	Set up another node as a Wi-Fi AP and TCP server	See also Use case 1: Wi-Fi local area network enabler
2	Configure the SPI slave to connect to the Wi-Fi network	See also Configuration (not stored in the module)

Once the module is connected to the network, use the following procedures to set up an SPI stream for incoming data and a subsequent TCP stream for forwarding this data on the network.

	Instructions	AT command
1	Set up an SPI stream on the mentioned pins with PDU size 720 and control protocol enabled.	`AT+UDCP="spi://spi0/?cs=32&sclk=31&miso=36&mosi=35&mode=3&drdy=25&size=720&proto=3”`
2	Set up a TCP stream to the Wi-Fi access point (check the IP address).	`AT+UDCP="tcp://192.168.2.1:8080"`
3	Bind the two streams together.	`AT+UDBIND=1,2`

Any data received from the SPI master is now forwarded over the TCP stream to the Wi-Fi access point.

4.7.11 Use case 11: Use secondary UART to send AT commands to a cellular modem

Supported modules	Software versions
ANNA-B412	All
NINA-B3	v3.0.0 onwards
NINA-B4	All

It is possible to configure a secondary UART stream. You can then use that stream to send commands to another module connected to the UART or use it as a data stream for example.

In this example, AT commands are sent to a u-blox cellular modem connected on this UART.

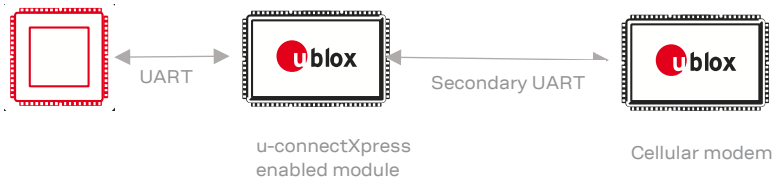


Figure 43 u-connectXpress enabled module connected to cellular modem using secondary UART

	Instructions	AT command
1	Set up the secondary UART. TX: GPIO_42 RX: GPIO_43 RTS: GPIO_5 CTS: GPIO_4	`AT+UMRSCFG=1,1,42,43,5,4`
2	Set up the UART stream to the modem. com2 = Secondary UART	`AT+UDCP="com://com2/?settings=115200,8,1,none,ctsrts&misc=true,0,500"`
3	Enter data mode. Any data sent over the primary UART is then passed to the secondary UART, so it is possible to send AT commands to the modem.	`ATO1`
4	Read the language on the SIM in the cellular module.	`AT+CLAN?` `+CLAN: "sv"`

☞ For information about the available GPIOs to which the secondary UART can be assigned, see the respective data sheet [21][22][23][24].

4.7.12 Use case 12: Data in AT mode over Bluetooth Low Energy

Supported modules	Software versions
ANNA-B112	v4.0.0 onwards
ANNA-B412	All
NINA-B1	v7.0.0 onwards
NINA-B2	v4.0.0 onwards
NINA-B31	v4.0.0 onwards
NINA-B41	v2.0.0 onwards
NINA-W13	v4.0.0 onwards
NINA-W15	v4.0.0 onwards

It is possible to send and receive data in text, hex, or binary format without entering Data Mode, in the example below, data will be send using u-blox Bluetooth LE Serial Port Service connection [31]. It can also be used for any peer connection created with a url-scheme starting with "at-". See also `+UDCP` [6].

4.7.12.1 Configuration

	Instruction to setup the first module (device 1) as Peripheral	AT command
1	**Device 1:** Enable the Peripheral Role.	`AT+UBTLE=2`
2	Store the configuration.	`AT&W`
3	Restart the device.	`AT+CPWROFF`
4	Set server configuration ID 1 to Serial Port Service. Not necessary on B1, B3 and B4.	`AT+UDSC=1,6`
5	Enable data in AT Command mode, <id> as given by `AT+UDSC` [6]. On B1, B3 and B4 <id> is 0	`AT+UDSF=<id>,2`
6	Store the configuration.	`AT&W`
7	Restart the device.	`AT+CPWROFF`

	Instruction to setup the second module (device 2) as Central	AT command
1	**Device 2:** Enable the Central Role.	`AT+UBTLE=1`
2	Store the configuration.	`AT&W`
3	Restart the device.	`AT+CPWROFF`
4	Connect peer using Serial Port Service. Use the address of Device 1.	`AT+UDCP="at-sps://D4CA6EB9227D"`
5	Wait for the URCs confirming peer connection.	`+UUBTACLC:0,0,D4CA6EB9227Dp` `+UUDPC:1,1,4,D4CA6EB9227Dp,20`

4.7.12.2 Sending and receiving data

Once the connection has been established.

	Instruction to send and receive data	AT command
1	**Device 2:** Check if data is available	`AT+UDATR=1,1,0` `+UUDATA:1,0`
2	**Device 1:** Send data “sentFromDevice1” using `+UDATW`	`AT+UDATW=1,0,"sentFromDevice1"`

3	**Device 2:** Check if data is available
	
	Data bytes available to read
		`AT+UDATR=1,1,0`
		`+UDATR:0,`
		`OK`
		
		`+UUDATA:1,15`

4	**Device 2:** Read 16 bytes in HEX mode	`AT+UDATR=1,1,15`
		
		`+UDATR:15,73656E7446726F6D44657669636531`

5	**Device 2:** Send data using `+UDATW`	`AT+UDATW=1,0,"sentFromDevice2"`

6	**Device 1:** Read 16 bytes in HEX mode	`AT+UDATR=1,1,15`
		
		`+UDATR:15,73656E7446726F6D44657669636532`

4.7.13 Use case 13: Data in AT mode using TCP sockets

Supported modules	Software versions
NINA-W13	v4.0.0 onwards
NINA-W15	v4.0.0 onwards

This example demonstrates how to use data in AT mode over TCP sockets.

4.7.13.1 Configuration

	Instruction	AT command
1	**Device 1:** Set up one device as an access point and TCP server on port 5003 on the local network, as described in Use case 2: Serial to Wi-Fi access point.	
2	**Device 2:** Connect to Device 1 in station mode using the steps described in Use case 1: Serial to Wi-Fi station.	
3	**Device 2:** Connect to the TCP server using an at-tcp scheme (IPv4 address `192.168.2.1` used in this example.	`AT+UDCP=at-tcp://192.168.2.1:5003`
4	**Device 2:** Wait for the URCs confirming peer connection.	`+UUDPC:1,2,0,192.168.2.100,49305,192.168.2.1,5003`
5	**Device 2:** Send some data	`AT+UDATW=1,0,"My text"`
6	**Device 1:** From Device 1 paste some text in the terminal	Paste `“Other text”`
7	**Device 2:** Event indicating data available	`+UUDATA:1,10`
8	**Device 2:** Send AT command to read data (done automatically in s-center)	`AT+UDATR=1,2,10`
9	**Device 2:** Event with data length read and actual data	`+UDATR:10` `Other text`

5 Optimization

☞ The performance optimization techniques may not be available for all software versions of all modules. For more information about AT commands necessary for module optimization, see also the u-connectXpress AT commands manual [6].

5.1 Wi-Fi optimization

- To improve the ping response, disable all low-power modes. The low power management is enabled by default and can be turned off using the AT command to get better response time and performance using Wi-Fi. Use the `AT+UWCFG=1,0` command.
- To optimize TCP connections for short latency, (especially for small data packet that improve the performance), activate TX flush.
  - For outgoing TCP connections, specify `<flush_tx=1>` in the URL in the `AT+UDCPC` or `AT+UDDRP` command. Example:
    `AT+UDCP="tcp://192.168.0.1:5003/?flush_tx=1"` or
    `AT+UDDRP=0,"tcp://192.168.0.1:5003/?flush_tx=1",2` for always connected.
    For incoming TCP connections, the Option 2 is set to 1 to enable TX flush on the listening port. Example: `AT+UDSC=1,1,5003,1`
- To increase throughput in cases with high degree of packet loss, increase the TCP out of sequence queue length. Use the `AT+UDCFG=5` command. Example:
  `AT+UDCFG=5,15`
- To increase the number of possible TCP links, decrease the TCP out of sequence queue length. Use the `AT+UDCFG=5` command. Example:
  `AT+UDCFG=5,0`

5.2 Bluetooth BR/EDR optimization

- For best performance, keep the allowed number of links as small as possible. Use the `AT+UBTCFG=1` command. The default is one link (only point-to-point).
- To maximize the throughput and minimize jitter on the data, the page and inquiry scan is turned off when link is connected. To change this, use the `AT+UBTCFG=6,0` command and `1` to disable this, (though this need not be done normally).
- To maximize the range, select only to use DM1 (one slot) packet using the `AT+UBTCFG=3,8` command. This lowers the throughput (to about 100 kbit/s) and improves the latency. This command works for both incoming and outgoing connections.
- Quality of Service (QoS) can be enabled for links where the module is the Central using the `AT+UBTCFG=5,1` command. This ensures that the shortest possible poll interval to the connected slaves is used.
- When it is required to get the lowest latest latency possible, the Active Poll configuration is recommended, and is enabled by `AT+UBTCFG=100,1`. This command should only be set on either the Central or the Peripheral, and not on both.
- To increase the throughput in noisy environments, enable the RFCOMM Enhanced re-transmission mode (ERTM). This improves the management of lost packets and decrease the number of undetectable bit-errors. Use the `AT+UBTCFG=12,1` to enable ERTM. Increase the MTU for the ERTM to further increase the throughput. Use the `AT+UBTCFG=13,1,1000` for maximum throughput.

5.3 Bluetooth Low Energy (LE) optimization

- To improve throughput, enable sending of LL PDU payload size (Data Length Extension) and ATT MTU size negotiation requests, using `AT+UBTLECFG=26,1`. This ensures that the highest possible MTU is used.

☞ Adjust the connection interval for optimal performance.

☞ To improve throughput using SPS, experiment with the connection intervals. The minimum value is 7.5 ms (6 * 1.25 ms) and is set using the `AT+UBTLECFG=4,6` and `AT+UBTLECFG=5,6` commands. For recommended connection intervals, see also the u-connectXpress Throughput Measurements application note [27].

☞ All remote devices do not support this low connection interval.

☞ For ANNA-B1, ANNA-B41, NINA-B1, NINA-B2, NINA-B31, NINA-B41 and NINA-W15, it is also required to enable sending of LL PDU payload size (Data Length Extension) and ATT MTU size negotiation requests.

- To decrease power consumption, use long connection intervals.

5.4 ODIN-W2 Wi-Fi and Bluetooth coexistence optimization

For information about Wi-Fi and Bluetooth coexistence, see the ODIN-W2 Bluetooth and Wi-Fi Coexistence application note [10].

5.5 Power consumption optimization

The major power consumer is the radio, especially when transmitting. Hence, the most efficient way to consume power, is to transmit as seldom as possible.

☞ The availability of the suggestions below, depends on the module and software.

Things that directly affect radio transmission frequency include Bluetooth LE advertising intervals, which is configurable with `AT+UBTLECFG=1` and `2`.

The amount of data to transmit at once also affects power consumption. Rather than sending 1 byte every ms to the module over the UART, it is more efficient to send 1000 bytes every second.

Increasing the MTU in RFCOMM Enhanced re-transmission mode (ERTM), by enabling ERTM with `AT+UBTCFG=12,1` and maximize the MTU with `AT+UBTCFG=13,1000`.

Increasing the Wi-Fi Station Listen interval, by using `AT+UWSC=<configuration_id>,300,<param_val>` to improve the efficiency of STANDBY and SLEEP mode.

Enable Automatic Frequency Adaption (AFA) using `AT+UPWRMNG` to decrease CPU and RAM power consumption during STANDBY and SLEEP mode.

Disable the UART by de-asserting the DTR line to improve efficiency of STANDBY and SLEEP modes. Re-enable the UART by asserting the DTR line. The UART is automatically re-enabled when an incoming Bluetooth SPS connection is established.

☞ To disable the UART, it is also necessary to change the circuit 108/2 (DTR) behavior. Change the behavior using the `AT&D3` command prior to de-asserting the DTR line.

To use the lowest power possible, enter STOP mode, either using `AT+USTOP` or change the circuit 108/2 (DTR) behavior using the `AT&D4` command, then de-assert the DTR line. This turns the module off completely. Resume operation by asserting the DTR line or the GPIO pin set with `AT+USTOP`.

☞ For software versions that do not support the `AT&D4` command, the alternative is to disable power to the module completely.

Appendix

A Glossary

Abbreviation	Definition
AES	Advanced Encryption Standard
AFA	Automatic Frequency Adaption
AFH	Adaptive Frequency Hopping
AP	Access Point
LE	Bluetooth Low Energy
BR/EDR	Basic Rate/Enhanced Data Rate
CCMP	Cipher Block Chaining Message Authentication Code Protocol
CTS	Clear To Send
DCE	Data Communication Equipment
DER	Distinguished Encoding Rules
DHCP	Dynamic Host Configuration Protocol
DNS	Domain name system
DSR	Data Set Ready
DTE	Data Terminal Equipment
DTR	Data Terminal Ready
DUN	Dial-up Networking Profile
EDM	Extended Data Mode
EVK	Evaluation Kit
GATT	Generic Attribute Profile
GPIO	General-purpose input/output
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IE	Information Element
IP	Internet Protocol
IoT	Internet-of-Things
JSON	JavaScript Object Notation
LAN	Local Area Network
MAC	Media Access Control
MCU	Micro-Controller Unit
MISO	Master Input, Slave Output
MOSI	Master Output, Slave Input
MQTT	Message Queuing Telemetry Transport
MQTT-SN	MQTT Sensor Network
NAP	Network Access Point
NFC	Near Field Communication
NTP	Network Time Protocol
OOB	Out of band
OKC	Opportunistic Key caching
OSI	Open Systems Interconnection model
OUI	Organizationally Unique Identifier

Abbreviation	Definition
PAN	Personal Area Networking
PANU	Personal Area Network User
PEM	Privacy Enhanced Mail
PHY	Physical Layer
PPP	Point to Point Protocol
RMII	Reduced media-independent interface
RTC	Real Time Clock
RTS	Request To Send
SiP	System in Package
SNTP	Simple Network Time Protocol
SPI	Serial Peripheral Interface
SPP	Serial Port Profile
SPS	Serial Port Service
SSL	Secure Socket Layer
SSP	Secure Simple Pairing
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
URC	Unsolicited result code
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VSIE	Vendor-specific Information Element
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access

Table 13: Explanation of the abbreviations and terms used

B Deprecated configurations

B.1 Bond two devices with Low Energy secure connections (old version)

Supported modules	Software versions
ANNA-B1	v2.0.0
NINA-B1	v5.0.0

Low energy secure connections is a feature that adds extra security to the bonding phase of the connection. In this example, two devices are bonded using the Numeric Comparison association model.

☞ The commands used for NINA-B1 SW 6.0.0, ANNA-B1 SW 3.0.0, and above have changed slightly. See also Use case 10: Change device information values.

	Instruction to setup the module	AT command
1	Device A+B: Enable Secure Connections in FIPS mode. This makes the device deny bonding with any device not supporting low energy secure connections.	`AT+UBTST=2`
2	Device A: Set Security mode 4 (Display YesNo)	`AT+UBTSM=4`
3	Device A: Set as Central	`AT+UBTLE=1`
4	Device A: Store and restart	`AT&W` `AT+CPWROFF`
5	Device B: Set Security mode 4 (Display YesNo)	`AT+UBTSM=4`
6	Device B: Set as Peripheral	`AT+UBTLE=2`
7	Device B: Store and restart	`AT&W` `AT+CPWROFF`
8	Device A: Initiate bonding with device B	`AT+UBTB=112233445566,1`
9	Device A+B: Note the passkey display event	`+UUBTACLC:0,0, <remote address>,` `+UUBTUPD: <remote address>,<passkey>`
10	Device A+B: Send response event indicating passkey displayed on the devices match.	`AT+UBTUPE=<remote address>,1`
11	Device A+B: Bonding event indicates successful bonding	`+UUBTB:<remote address>,0`

Bonding is now completed.

If one of the devices does not support low energy secure connections (`AT+UBTST=0`) the bonding is denied.

Related documents

[1] https://github.com/u-blox
[2] https://www.mbed.com
[3] Evaluation kit for ODIN-W2 series user guide, UBX-15020900
[4] ODIN-W2 series system integration manual, UBX-14040040
[5] NINA-W1 series system integration manual, UBX-17005730
[6] u-connectXpress AT commands manual, UBX-14044127
[7] https://www.bluetooth.com/specifications/gatt/services
[8] u-blox Extended data mode protocol specification, UBX-14044126
[9] u-blox Bluetooth security application note, UBX-16022676
[10] ODIN-W2 Bluetooth and Wi-Fi coexistence application note, UBX-18021138
[11] NINA-B1 series system integration manual, UBX-15026175
[12] ANNA-B112 system integration manual, UBX-1800982
[13] NINA-B2 series system integration manual, UBX-18011096
[14] NINA-B3 series system integration manual, UBX-17056748
[15] https://www.bluetooth.com/specifications/gatt/services
[16] https://www.bluetooth.com/specifications/gatt/characteristics
[17] https://developer.apple.com/ibeacon/
[18] https://github.com/google/eddystone
[19] u-connectXpress MQTT application note, UBX-19005066
[20] u-connectXpress IoT Cloud connectivity application note, UBX-19010078
[21] ANNA-B112 data sheet, UBX-18011707
[22] NINA-B1 series data sheet, UBX-15019243
[23] NINA-B2 series data sheet, UBX-18006649
[24] NINA-B3 series data sheet, UBX-17052099
[25] NINA-B41 series data sheet, UBX-20035327
[26] ANNA-B412 data sheet, UBX-21028698
[27] NINA-W13 series data sheet, UBX-17006694
[28] NINA-W15 series data sheet, UBX-18006647
[29] ODIN-W2 series data sheet, UBX-14039949
[30] u-connectXpress Throughput measurements application note, UBX-17023548
[31] u-blox Low Energy Serial Port Service, UBX-16011192
[32] Communicating with a u-blox module over SPI bus application note, UBX-20028725
[33] https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-mobile
[34] u-connectXpress Wi-Fi security application note, UBX-20012830

☞ For product change notifications and regular updates of u-blox documentation, register on our website, www.u-blox.com.

Revision history

Revision	Date	Name	Comments
R01	25-Aug-2017	cmag	Initial release.
R02	22-Dec-2017	cmag, kgom	Updated the applicable products table on page 2 to include support for ODIN-W2-SW 5.0.0. (Use the ODIN-W2 Getting started (UBX-15017452) for the ODIN-W2 software versions 1.0.0 to 4.0.1). Added information about Wi-Fi Roaming (section 3.7) and Bridge functionality (section 3.8). Also added information about the following use cases - Bluetooth Personal Area Network (section 4.4.8), Wi-Fi AP and Bluetooth PAN NAP bridge (section 4.4.9) and Wi-Fi Station connecting to Enterprise security using EAPTLS (section 4.7.4).
R03	26-Jan-2018	kgom	Included support for ODIN-W2 software version 5.0.1.
R04	19-Apr-2018	mhan	Updated configuration for an example in Ethernet to Wi-Fi Bridge use case (section 4.1.5).
R05	20-Jun-2018	cmag, kgom	Updated the applicable products table on page 2 to include support for ODIN-W2-SW 6.0.0. Included information about Bind functionality (section 3.10). Updated Wi-Fi roaming with threshold value (section 3.7).
R06	24-Sep-2018	cmag	Updated Bridge functionality with additional examples (section 3.8). Made a minor change in sections 3.10 and 4.3.1.1. Updated sections 4.1, 4.1.6, and 4.4.
R07	17-Dec-2018	cmag, kgom	Made this document generic for more u-blox short range stand-alone modules such as NINA-W13 in addition to ODIN-W2.
R08	6-Feb-2019	cmag, mape	Added NINA-B1, ANNA-B112, NINA-B2, and NINA-B31 modules to the Bluetooth use cases.
R09	5-Mar-2019	fbro, mape,	Replaced "u-blox connectivity software" with "u-connectXpress software" in all instances. Added support for NINA-B316, NINA-B1, and ANNA-B1 SW 2.0.0. Modified the document type as “User Guide”.
R10	19-Mar-2019	cmag, kgom	Updated ODIN-W2 Wi-Fi Roaming (section 3.7) and Example of a Bridge Configuration without the DHCP server (section 3.8.1). Included a note in Bind functionality (section 3.10). Added information about TLS (section 3.9) and MQTT (section 3.11) and use cases for the same. Included information about Certificate upload in use case 4 (section 4.7.4).
R11	28-Jun-2019	cmag, mape	Included support for NINA-W15. Changed low energy secure connections example to use Numeric Comparison.
R12	30-Oct-219	flun, mape	Updated section 3.2 with references to LEDs. Updated the Related documents section. Corrected use-cases 4.1.4 Serial PPP to Wi-Fi Station, 4.6.2 Wi-Fi access point to Serial PPP and section 3.2.4 PPP mode. Added section 4.5.6 Use case #6. Updated CODED PHY connection in section 4.5.7. Updated section 4.1.5 RMII/Ethernet to Wi-Fi Station Bridge. Added sections 4.6.11 and 4.6.12 for new use-cases. Added support for NINA-W13 to use-cases Reading and writing GPIO pins (section 4.6.7), Connect using TLS (section 4.6.8), and MQTT Client Gateway (section 4.6.10). Added several suggestions for optimization of Wi-Fi (section 5.1), Bluetooth BR/EDR (section 5.2), Bluetooth Low Energy (section 5.3), and power consumption (section 5.4). Moved ODIN-W2 Wi-Fi and Bluetooth coexistence to a separate section (section 5.5).
R13	19-Nov-219	cmag	Added product variant ODIN-W263.
R14	12-Feb-2020	mape	Added explanation about connection handles and peer handles in 3.4
R15	16-Mar-2020	mape, flun, ctur	Added new section to describe system control signals and included editorial updates in several sections.

Revision	Date	Name	Comments
R16	10-Jul-2020	flun, mape,	Clarified TCP peers, options, added mqtt scheme in section 3.5.1: Added examples for TLS in section 3.5.2. Added new section (3.5.6) to describe MQTT peers. Added new section (3.12.4) to describe IoT security. Moved IoT use-cases to new section (0). Added sections 4.6.4 to 4.6.6 to describe connections to an IBM Watson IoT Platform, Amazon AWS IoT Core, and Microsoft Azure IoT Hub. Added MQTT Client Gateway: Datamode supports subscription of subtopics. Added new commands/events for LE Secure Connections in section 4.5.10. Added example with multiple central connections in section 4.5.12. Moved BLE-specific cable replacement to Bluetooth LE section 4.5.15 and 4.5.16. Added ac-to parameter to AT+UDDRP use cases in sections 4.1.1, 4.4.2, and 4.5.16. Added SPI stream example in section 4.7.10. Updated Bridge example in section 3.7.2. Updated glossary section. Added chapters 4.5.15 (automatic PHY adaptation) and 4.7.11 (Using secondary UART). Added new use case to section 4.5.17 to describe how to connect two modules and use automatic PHY adaptation. Added new use case to section 4.7.1 to describe how to configure the bridge the module to route all Layer 2 traffic between the Wi-Fi AP interface and an Ethernet interface.
R17	15-Jul-2020	flun	Added new chapter 3.3 on low power modes and extended chapter 5.5 on power consumption optimizations. Minor editorial changes to chapter 3.2 on low power modes and chapter 5.1 on Wi-Fi optimizations. Added DTR, SPI_CS, SPI_CLK and RMII_CLK to chapter 3.4.1 Added DRS and DRDY to chapter 3.4.2. Added HTTP-TCP peer in chapter 3.6.7. Added SPI peer in chapter 3.6.8 and 3.10. Added HTTP, HTTPS and NTP clients and AWS qualified logotype to chapter 3.12. Added use cases for GET and POST of JSON using HTTP/HTTPS in chapter 4.6.7 and 4.6.8. Added use cases for system time and NTP time in chapters 4.6.9 and 4.6.10 Deprecated use case “Bond two devices with Low Energy secure connections (old version)” moved to sub-chapter of Deprecated use cases in Appendix B.
R18	1-Sep-2020	cmag, flun	Minor editorial changes: Updated example SSID and passwords for consistency, changed size and removed some punctuation. Added note on Bluetooth impersonation attacks to section 3.13.3. Added example on how to create a HTTPS connection directly in a TLS stream to a web server in section 4.6.1.
R19	30-Oct-2020	mape	Added NINA-B41x to examples.
R20	23-Nov-2020	mape	Corrected pin numbers used in use case #11 in section 4.7.
R21	22-Jan-2021	mape	Changed pin numbers used in use case #11 in section 4.7 to non-radio sensitive pins on NINA-B3 and NINA-B4. Revised terminology to avoid discriminatory language wherever possible (except references to existing code or program output). Added Use case #3: Letting the system handle GATT characteristic values and Use case #4: Long GATT writes. Added missing peers to section 3.6. Added NINA-W156 as an applicable product.
R22	1-Jun-2021	mape	Extended document scope to include NINA-B411 and included editorial changes in all sections.

Revision	Date	Name	Comments
R23	12-Aug-2021	ldas	Added +UDATW, +UDATR and IRK examples in Use case 18: Connect to random resolvable address device using Identity Resolving Key (IRK). Revised all document cross-references.
R24	1-Sep-2021	cmag	Added Bond with fixed key (headless pairing) using Bluetooth Low Energy use case.
R25	29-Oct-2021	mape	Extended document scope to include ANNA-B412. Included minor terminology changes in Use case 14: Bond with fixed pin (headless pairing) using Bluetooth Low Energy.
R26	25-Jan-2022	mape	Added NINA-B41 to use cases Use case 5: NFC links, Use case 6: Over the air configuration and Use case 7: Read and write GPIO pins. Minor corrections.
R27	11-Aug-2023	mape	Amended description of reset of serial port settings in Switches and input signals. Added chapter Use case 6: UDP connectivity.
R28	02-Apr-2024	mape	Added Use case 13: Data in AT mode using TCP sockets. Added description of Bluetooth addresses in Peers. Clarified that NINA-W15 and NINA-B2 needs to have Bluetooth low energy enabled in Use case 1: Set up a GATT server / client, Use case 3: Letting the system handle GATT characteristic values, Use case 4: Long GATT writes, Use case 5: Set up the modules as beacons. Removed deprecated use case Bluetooth low energy headless pairing. Added missing information about phy update event in Use case 8: Connect two modules and automatically switch to 2 Mbit/s PHY. Corrected setting of UBTLECFG tag 29 to only be valid for Peripheral in Use case 8: Connect two modules and automatically switch to 2 Mbit/s PHY. Enabled DHCP server in Use case 1: Wi-Fi local area network enabler and Use case 2: (Hosted) Wi-Fi tethering (hot spot). Corrected pin used to exit data mode in Figure 15.

Contact

u-blox AG

Address: Zürcherstrasse 68
8800 Thalwil
Switzerland

For further support and contact information, visit us at www.u-blox.com/support.