CELLULAR

NB-IoT Application Development Guide

Technology architecture and AT command examples

Application Note

Abstract

This document provides detailed examples of how to use AT commands with u-blox NB-IoT cellular modules.

www.u-blox.com

UBX-16017368 - R02

Document Information		
Title	**NB-IoT Application Development Guide**	
Subtitle	Technology architecture and AT command examples	
Document type	Application Note	
Document number	UBX-16017368	
Revision, date	R02	02-Feb-2017
Disclosure restriction	Confidential	

This document applies to the following products:

Product name	FW version
SARA-N2 series	V100R100C10B650SP11
	V100R100C10B650SP8

Contents

1 Introduction

This document provides guidance when developing applications for NB-IoT. It includes examples of using AT commands to communicate with the current u-blox NB-IoT and how to send UDP packets. See the SARA-N2 series AT Commands Manual [2] for detailed AT command descriptions.

The following symbols are used to highlight important information within this document:

☞ An index finger points out key information pertaining to module integration and performance.

⚠ **A warning symbol indicates actions that could negatively impact or damage the module.**

2 NB-IoT technology overview

NB-IoT technology is designed such that it can be used in areas beyond the radio coverage of current cellular standards and in devices which must run from battery power for many years. A corollary of this is that the devices will generally send small amounts of data infrequently; a typical usage scenario might be 100 bytes sent twice per day.

The system operation is analogous to SMS in that it is a datagram-oriented, stored-and-forward system, rather than a GPRS-like IP pipe. This is because NB-IoT devices spend most of their time asleep, making possible the required long battery life. The system implements extended DRX cycles for paging, but as this window will be limited to save battery life, the delivery of downlink messages occurs mainly when the system detects that uplink messages have been received from a device (indicating that it is awake). Here a store-and-forward system, an "IoT Platform", is useful.

Simplistically, the system can be represented as shown in Figure 1.

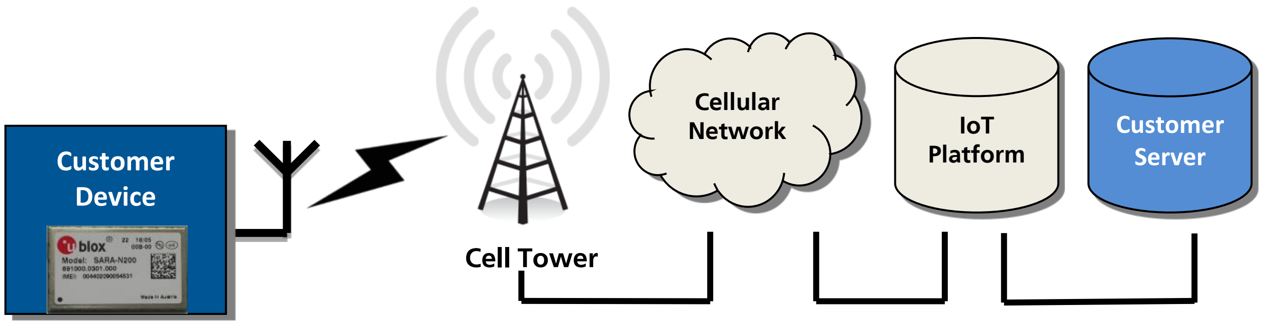


Figure 1: NB-IoT system architecture

At the far left the customer's device contains a u-blox NB-IoT module that communicates over the radio network with a cell tower that supports NB-IoT. The cellular network links the cell tower with an IoT platform. This IoT platform stores uplink datagrams from the NB-IoT module. The customer server communicates with the IoT platform to retrieve uplink datagrams and to send downlink datagrams to the NB-IoT. The IoT platform holds downlink datagrams until the NB-IoT module is awake to receive them.

Currently the SARA-N2 series modules implement basic UDP socket commands for directly communicating with an external service. With these commands the customer can build a simple IoT Platform. With an external processor CoAP and other IoT layers could be implemented to aid this design of system.

☞ Huawei have their own IoT platform, with dedicated AT commands, but this platform may not be available to the customer. See your network operator for options on IoT platforms.

3 AT command response parser

This section gives some hints about how to develop an AT parser and how to handle the AT command replies and the URCs (unsolicited result code).

In this document the following naming conventions are used:

- DCE (Data Communications Equipment) or MT (Mobile Terminal) is the u-blox NB-IoT module
- DTE (Data Terminal Equipment) or TE (Terminal Equipment) is the terminal that sends the command to the module

When entering AT commands, spaces are ignored. The DCE uses carriage-return line-feed pairs (`\r\n`, `0x0D0A`) to end lines on its output. The same termination is required on input to the DCE.

When the DCE has finished processing a line it will output either `OK` or `ERROR` indicating that it is ready to accept a new command. Solicited informational responses are sent before the final `OK` or `ERROR`.

3.1 Unsolicited Result Code

An unsolicited result code (URC) is a string message (provided by the DCE) that is not a response to a previous AT command. It can be output, when enabled, at any time to inform the DTE of a specific event or status change. The implemented URCs are as follows:

• +CEREG: <stat>[,<tac>,<ci>,<AcT>]	Registration
• +NPING:<retry_num>,<remote_address>,<ttl>,<rtt>	Ping
• +NSONMI:<socket>,<length>	Received data on socket
• +NNMI:<length>,<data>	New message indicator
• +NNMI	New message indicator
• +NSMI:SENT	Sent message indicator

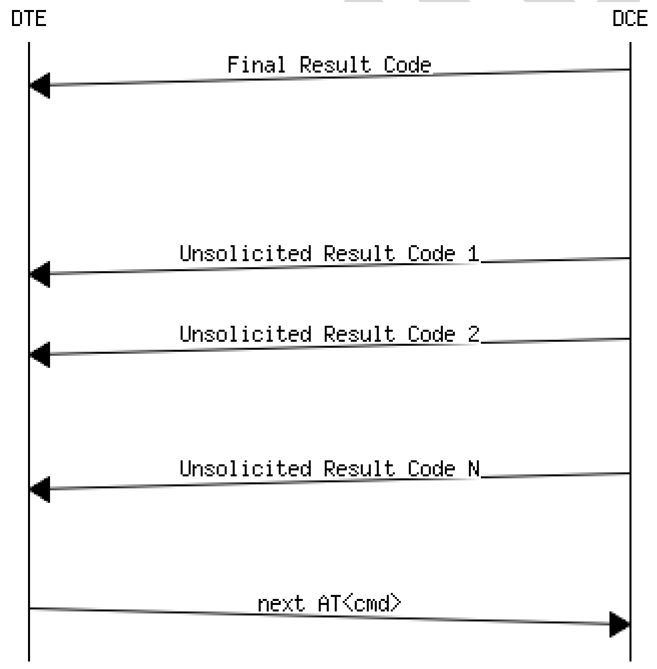

DTE
DCE
Final Result Code
Unsolicited Result Code 1
Unsolicited Result Code 2
Unsolicited Result Code N
next AT<cmd>

Figure 2: DTE-DCE URC flow chart

3.2 Best practices

- The DTE shall flush the AT channel (i.e. check if there is data waiting to be read) before sending a new AT command.
- The DTE shall handle the case of unexpected spaces or line endings.
- The DTE shall handle all the URCs: it can simply ignore them (not suggested) or, better, take a proper action.
- The DTE shall know what answer is expected and shall wait until it is received (i.e. final result code only or information text response with the final result code).
- The final result code marks the end of an AT command and can be OK or ERROR. When the final result is an error, be sure to handle it before continuing with the next AT command.
- The information text response format is command specific. The DTE will need explicit handling for each one. It is suggested to consult the SARA-N2 AT Command Manual [2].
- It is suggested to not strictly parse information text responses but rather to check if they contain interesting keywords and/or parameters.
- It is very useful, for debugging an application, to log all the command lines sent to the DCE and received from it.
- Create a state machine for the AT parser (e.g. idle, waiting_response, data_mode).
- The DTE shall wait some time (the recommended value is at least 20 ms) after the reception of an AT command final response or URC before issuing a new AT command to give the module the opportunity to transmit the buffered URCs. Otherwise the collision of the URCs with the subsequent AT command is possible.

4 Registration with the NB-IoT network

By default SARA-N2 series modules will try and automatically connect to a network using the present SIM. This can be turned off by the +NCONFIG AT command. If this feature is turned off, the following AT commands will allow the application to connect to a network.

4.1 Manually starting registration process

1. Turn on the module's radio functionality

Command	Response	Description
`AT+CFUN=1`	`OK`	Enable full radio functionality.

2. Start the registration process and attach to the Packet Domain Service. This will read the PLMN from the attached SIM.

Command	Response	Description
`AT+CGATT=1`	`OK`	Configure the PDP context.

☞ The +CGATT AT command will create a context ID starting from zero (0).

4.2 Signaling connection status

Issue the AT+CSCON read command to poll the base station connection status. This connection is only active when transmitting or receiving data.

Command	Response	Description

`AT+CSCON?`	`+CSCON:<mode>,<status>`
	`OK`
		Allowed <status> connection status:
		• <status>: 0 – Idle
		• <status>: 1 – Connected

4.3 Registration status

Before sending any data to the NB-IoT network wait for the device to be registered to a network. Issue the AT+CEREG read command to poll the status. It is possible to enable a URC which will automatically output any change to this status.

Command	Response	Description

`AT+CEREG?`	`+CEREG:<mode>,<status>`
	`OK`
		Allowed <status> connection status:
		• 0: not connected
		• …
		• 5: registered, roaming

4.4 Obtaining IP address

Before sending any data to the NB-IoT network wait for the device to be configured with an IP address. Issue the +CGPADDR AT command to see if an IP address has been set.

Command	Response	Description

`AT+CGPADDR=<cid>`	`+CGPADDR:<cid>,`
	`OK`
	
	`+CGPADDR:<cid>,<ip address>`
	`OK`
		There is no IP address yet.
		
		The module has an IP address.

5 UDP sockets

The SARA-N2 series modules are able to send raw data through UDP sockets to an IP address. The data sent over the socket AT commands is not wrapped in any other layer, and the data provided is the data that is sent.

☞ If the module is not being used with the Neul or Huawei IoT platform, this is the only other method of sending and receiving data over an NB-IoT network.

5.1 Creating a socket

Create a socket to be able to send UDP data. A socket ID is returned.

Command	Response	Description

`AT+NSOCR=DGRAM,17,<localport>`	`<socketID>`
	`OK`
		Create a socket with a listening port.
		Returns the socket ID to be used with other socket commands.

5.2 Closing a socket

Once a socket is no longer needed, it should be closed.

Command	Response	Description
`AT+NSOCL=<socketID>`	`OK`	Specify the socket ID to close.

5.3 Sending UDP data

Sending data to an external server is as simple as specifying the socket to use, the remote IP address and port, and then the length of data, plus the data. The information text response provides the number of bytes successfully sent.

☞ The maximum length of data that can be sent is 512 bytes.

Command	Response	Description
`AT+NSOST=<socketId>,<remote_address>,<remote_port>,<length>,<data>`	`<socketId>,<length>`	Send data to the specified IP address and port through the socket noted by the ID.

5.4 Receiving UDP data

Receiving data is performed in two steps. If the module has received data from the network on a socket that is listening, then a URC is given. From this message, the application can read the data on the appropriate socket and length.

5.4.1 Data arrived indicator

Command	Response	Description
	`+NSONMI:<socketId>,<length>`	This message is provided to tell the application how much data is available to read on the specified socket.

The application should read this message and then read the data from the specified socket.

5.4.2 Reading data from socket

The data reception is performed by means of the +NSORF command, using the information given in the +NSONMI URC.

Command	Response	Description

`+NSORF:<socketId>,<length>`	`<socket>,<ip_addr>,<port>,<length>,<data>,<remaining>`
	`OK`
		Provides the received data to the application and shows how much data is still left to read out.

5.5 Testing UDP sockets

A simple way to test UDP sockets over NB-IoT is to send data to an echo server.

☞ u-blox echo server: echo.u-blox.com.

Here is an example:

Command	Response	Description

`AT+NSOCR=17,14000`	`1`
	`OK`
		Create a socket. The socket ID is 1.
		Ready for next AT command

`AT+NSOST=1,195.34.89.241,7,5,48656c6c6F`	`1,5`
	`OK`
		Send "Hello" to u-blox echo server. Sent 5 bytes on socket ID 1.

	`+NSONMI:1,5`	Received 5 bytes on socket ID 1

`AT+NSORF=1,5`	`1,195.34.89.241,7,5,48656c6c6F,0`
	`OK`
		Read 5 bytes on socket ID 1.
		Received data information provided…"Hello".

6 Checking module statistics

Command	Response	Description

`AT+NUESTATS?`	`Signal power:<number>`
	`Total power:<number>`
	`TX power:<number>`
	`TX time:<number>`
	`RX time:<number>`
	`Cell ID:<number>`
	`DL MCS:<number>`
	`UL MCS:<number>`
	`DCI MCS:<number>`
	`ECL: <number>`
	`OK`
		The response indicates the condition of the module and its environment. The values may be useful for monitoring purposes.

The signal power is the power of the wanted part of the receive signal, the NB-IoT part, whereas the total power is the radio signal strength within the receive bandwidth (both expressed in 10ths of a decibel). From this the signal to noise ratio can be calculated.

The transmit power is the RF power output from the module. It may be a low number if the received signal strength is good (and hence the module assumes that the base station is close by).

The Tx time is the duration for which the module's transmitter has been switched on and, likewise, the Rx time is the duration for which the module's receiver has been monitored for downlink activity (both expressed in milliseconds since the last reboot). Together these can be used to assess the time the module spends in each state and hence estimate the power consumed by the module.

The Cell ID is the physical cell ID of the cell that is currently providing service to the module.

The DL and UL MCS classes indicate the choices that the module/network has made concerning the coding of uplink and downlink messages. From the DL MCS the coverage class can be derived as follows:

ECL is equivalent to PRACH coverage enhancement level defined in 3GPP 36.321 [3] sub clause 5.1

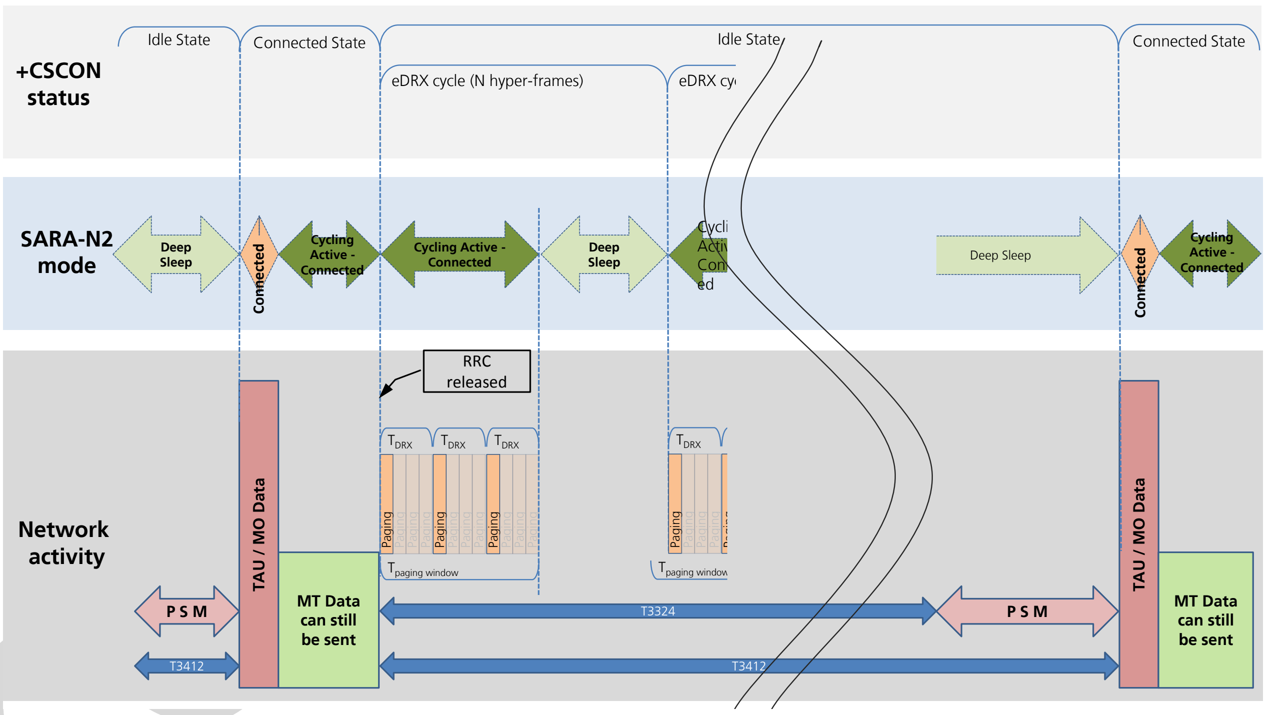
7 Paging, eDRX, Power Save Mode and Deep Sleep

The NB-IoT protocol allows for power save mode (PSM), and the SARA-N2 series modules also support a Deep Sleep mode where the module is running at very low current, ~5 µA. The module automatically enters various states depending on the device activity. Here below are listed the common activities and the various states it will be in after registration.

1. The device is in Power Save Mode (PSM) and in Deep Sleep; it is already registered and has nothing to do.
2. A message is queued or the TAU timer has elapsed.
3. Device re-connects to network and sends and receives data. This is in Connected State.
4. RRC connection is released by the network. It is now in Idle State.
5. Within T3324 timer, Paging happens as per Network configuration.
6. Power Saved Mode is entered after T3324 has elapsed.
7. T3412 elapses and Traffic Area Update is triggered, or application sends more data.

The module enters Deep Sleep mode when there is no activity, it does not need to be in PSM.

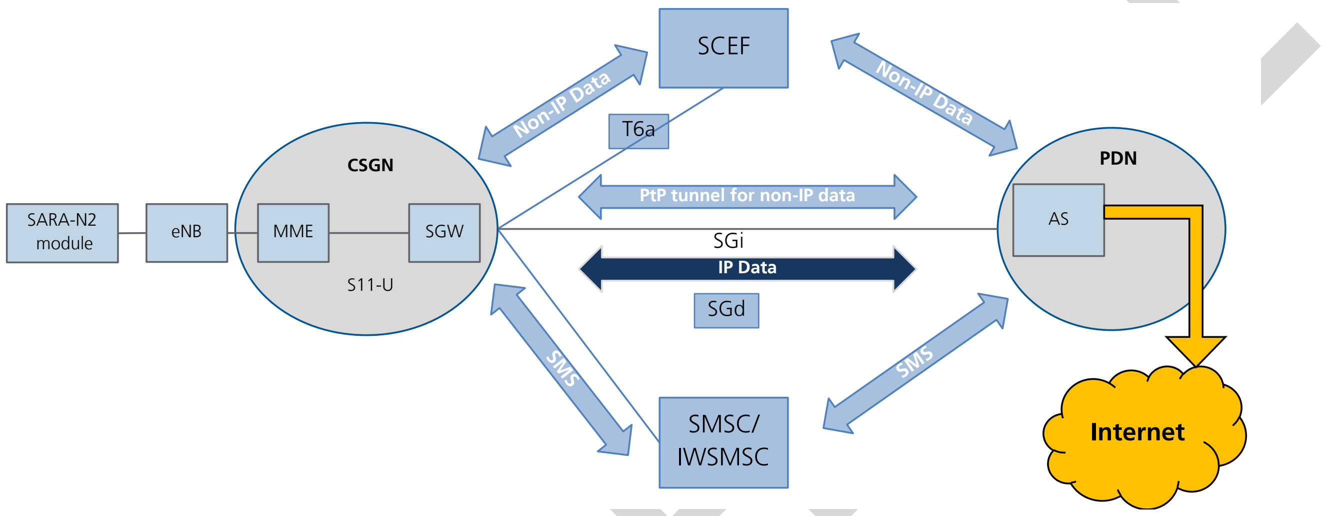
☞ T3324 and T3412 timers are assigned by the network and cannot be changed by the application.

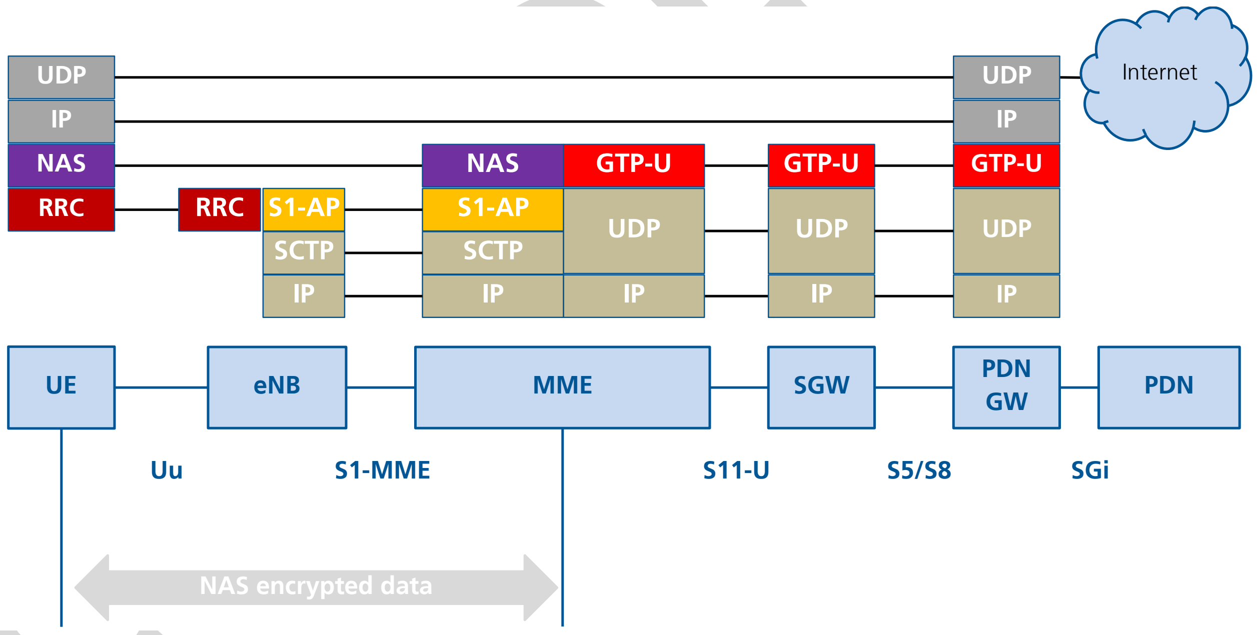


8 Network architecture mapping on AT commands

Currently NB-IoT networks and module firmware only supports IP transport. Non-IP and SMS are transports may be available later.

Both UDP socket commands and datagram commands use the IP data transport through the SGi.





9 NB-IoT best practices

9.1 Radio and network status polling

The application can monitor the radio and network status. Without registration or an IP address the application will not be able to successfully send or receive data.

- **AT+CEREG**: to query the registration status to the network.
- **AT+CGPADDR**: to query if the module has an IP address.
- **AT+CSCON**: to query the connection status to the base station. This is only useful to see if Tx or Rx activity is happening. It should not be used to decide whether or not to send data.

9.2 Deep-sleep mode

SARA-N2 series modules are designed to enter a power save mode which allows batteries to last longer. After the module connects to a base station to send a message, the module will stay connected to a base station for a period of time after the last communication with the base station. The device will then go back into deep-sleep mode.

An application that sends lots of messages throughout the day will obviously keep the module awake and connected to a base station, consuming power.

In an ideal design the application should not require the device to send more messages than necessary. The cloud application should not need to send the module many messages either, although in a trial system the customer may want more activity to see the operation of the system. In these cases high capacity batteries or continuous power supply should be used.

9.3 Message size

The module has a limited dynamic message queue size. For IoT applications, the message size should be of the order of tens of bytes. UDP socket commands limit their payload size to 512 bytes. Currently there is no indication when the UDP data has been sent.

9.4 Application architecture

Many developers coming from a GPRS type background may expect an always on type connection, normally using TCP. NB-IoT is not session oriented, the latencies between each packet and uplink/downlink are much higher and it is better that the device enters the sleep mode, rather than waste power using a “chatty” protocol. It is much better for the product to publish the data on a timely basis into the Cloud Service or IoT Platform using the UDP socket AT commands.

UDP sockets do not create connections to servers; UDP is a connection-less datagram protocol. MO messages may also not be received by the server – as there are no acknowledgements between the client and server. The application should take this in to consideration.

For resolving the issue of sending MT messages to a very sleepy module, when a MO message is sent to the cloud server, the cloud server will know the module is active and connected to the network. As seen in section 7 the connection is alive until the RRC connection is released by the network. If there are MT messages to be sent to the module, the cloud server should do this the moment a MO message is received.

☞ MT messages will be lost if the cloud service sends an MT message at other times. Only when the MO message has been received does the cloud server have a limited time to send a MT message.

10 Application examples

The pseudo code below shows some simple application functions.

10.1 Wait for registration

A basic function of an application would be to check if the module is actually registered to a network.

```
Public Success WaitForRegistration() {
      Do
            Registration = RequestAT("+CEREG?");
            Sleep(1);
      Loop until (Registration == 1 || Registration == 5 || TimedOut)
      Return !TimedOut;
}
```

10.2 Wait for IP Address

The module will receive an IP address after it has registered to the network, although this will be after some time, therefore the application should wait until it has one.

```
Public IPAddr WaitForIPAddress(cid) {
      Do
            IPAddr = RequestAT("+CGDPADDR=<cid>");
            Sleep(1);
      Loop until (IPAddr != empty || TimedOut)
      Return IPAddr;
}
```

10.3 Send UDP data

Sending data to a UDP socket is very simple. It is possible to check the length of the data sent was the same as queued.

```
Public SentLength SendUDP(socket, ipaddr, port, data[]) {
 Length = data.Length();
 Response = RequestAT("+NSOST=<socket>,<ipaddr>,<port>,<length>,<data>);
 Return Response[2];// sent length is the 2nd comma delimited string element.
}
```

10.4 Receive UDP data

After receiving the +NSONMI URC, it is possible to request for the data

```
Public data[] ReceiveUDP(socket, length) {
 Response = RequestAT("+NSORF=<socket>,<length>);
 Return Response[5];// data is in the 5th comma delimited string element.
}
```

11 Frequently Asked Questions

1. The module never connects to the base station (+CSCON Not Connected, +CEREG Not Registered, +CGPADDR No IP Address)
   - Check that the module is operating in the correct band (+RFBAND?)
   - Check that the module's IMEI is correctly set (+CGSN=1)
   - Check that the SIM is correctly inserted. (+CIMI)

2. The module is connected (+CSCON Connected, +CEREG Registered), but no messages can be sent (+NQMGS: Queued x, Sent 0)
   - Check that the module has an IP address (+CGPADDR). The core network may not allow that module on the network.
   - If using the Huawei IoT Platfrom, check that the +NCDP IP address is correctly set.

3. The module continuously resets when trying to connect to the base station
   - Check that the power supply is able to cope with the current consumption requirements of the module (see SARA-N2 series Data Sheet [1]). If the device is battery powered, check that the battery pack is not discharged.
   - Check the antenna matching. Mismatched antennas will consume more current during transmission bursts.

12 Application scenarios

12.1 Fit for NB-IoT

Module initiated, sparse, short messaging.

Depending on battery size, these types of applications are valid:

Events	Metering	Location tracking
• Parking	• Water	• Containers
• Security	• Gas	• Vehicles
• Waste	• Electricity	• Bikes
• Leakage	• Environment	
• Pest Control		
• Refill		

12.2 Not fit for NB-IoT

Server initiated messaging, very large messages and many messages per hour are types of applications which do not fit well with NB-IoT modules that are designed to be in a deep-sleep mode most of the time.

Immediate and continuous types of applications such as these are not valid:

Events	Metering	Tracking
• Emergency	• Continuous environment monitoring	• "Live" tracking of vehicles
• Health Care	• Feedback loops	
• Fire	• Surveillance	
• Flood		

13 Datagram messages with Huawei IoT platform

Both Huawei and Neul have compatible IoT platforms that allow datagram messaging to easily be sent and received. Dedicated AT commands have been implemented to interact with these IoT Platforms.

These IoT platforms are not always available. Please consult the network operator if this platform has been installed.

For both sending and receiving datagrams, the AT command format is similar:

```
<length>, <data>;
```

where `<length>` is the number of hex bytes that follows and `<data>` is the hex bytes. For example:

```
AT+NMGS=10,0102030405060708090A
```

This command would send a datagram containing the binary values 1 to 10 to the network as a single datagram. A datagram should not be longer than 1024 hexadecimal bytes.

☞ Datagrams sent and received by these commands are wrapped internally in a Constrained Application Protocol (CoAP) message and sent over UDP sockets to the IoT Platform. Although these commands use CoAP, these can only work with the Neul or Huawei IoT platform, not any CoAP based IoT Platform.

13.1 Sending MO datagrams

Command	Response	Description
AT+NMGS=<number>,<data>	+NMGS:OK OK	This response indicates that the module has accepted the datagram. If AT+NSMI=1 has been set, then the URC +NSMI:SENT will be issued when the datagram has been acknowledge by the base station.

☞ Once the module accepted a datagram it cannot be removed and will be transmitted to the network as soon as radio conditions permit. The only way to clear the module's transmit queue is to reboot it. In good radio conditions, the transmission might take a few seconds. In bad radio conditions a transmission opportunity may not occur for minutes, days or weeks but the datagram will be transmitted once radio conditions are good enough.

When a MO message is queued, the module will try to send the message to the base station. It will only send the next message once the previous message has been sent. If there is a radio link failure (RLF), the device will re-scan the channel ranges and try to reconnect to a base station. There may be a back off time where the device goes into deep-sleep mode before trying again.

13.2 Querying MO datagram sent status

Command	Response	Description
AT+NQMGS	PENDING=1,SENT=4,ERROR=0 OK	This response indicates how many datagrams have been queued, sent and not sent.

Unlike the +NSMI:SENT indication, which shows when the datagram has been sent to the base station, there is no indication if the core network or CDP server has received the message. The customer will need to implement an application based reply if notification is required to be given back to the device; however this is not best practice for IoT type products.

13.3 Receiving MT datagrams

To send an MT message, queue an MT message on the CDP server. Use the AMQP interface to queue a message on the server. See the example source codes (section 10.3) for more details.

The device will need to be registered on the CDP server on a user account. The CDP will provide an AMQP password to use, and can be viewed on the "My Account" page.

MT messages may not arrive at the module at the time of queuing on the CDP server, as the module is normally in deep-sleep mode. Only when the module exits deep-sleep mode and connects to a base station will the system be able to send the MT messages.

Modules can be configured on the CDP to periodically wake up (Heartbeat) and connect to a base station so that there is a chance for MT messages to be sent.

Command	Response	Description

`AT+NMGR`	`+NMGR:<number>,<data>`
	`OK`
		This response contains a downlink datagram that has been received by the module. The datagram is now removed from the module's queue. A datagram will not be longer than 1024 hex bytes.

`AT+NMGR`	`OK`	This indicates that no downlink datagrams are present in the module's queue.

One `+NMGR` response containing one datagram is issued for each `AT+NMGR`. To check for multiple downlink datagrams, multiple `AT+MGR` commands must be issued. There is no indication to the CDP server if the module received the datagram.

If the +NNMI URC has been enabled (see section 13.7), the module will send an unsolicited response on the UART after receiving a MT datagram. If the module's application, on customer device, is not prepared to read the message as it is presented on the UART interface, the message will be lost.

Command	Response	Description
	`+NNMI:<number>,<data>`	This response contains a downlink datagram that has been received by the module. The datagram is removed from the module's queue.

13.4 Querying MT datagram received status

Command	Response	Description

`AT+NQMGR`	`BUFFERED=0,RECEIVED=0,DROPPED=0`
	`OK`
		This response indicates how many datagrams have been received, queued and not received.

13.5 Avoiding data loss

While the NB-IoT system is built to avoid data loss due to RF signal issues, buffers are of a finite length and hence it is necessary to design with the limitations in mind.

In the uplink direction, it is recommended that the application on the device sets up send confirmations (see section 13.6) and only puts another datagram into the module when the previous datagram has been confirmed as sent. This has the added advantage that the send buffer can be managed by the application (so, for instance, stale data is not left queued in the module).

In the downlink direction, it is recommended that the customer server does not allow datagrams to build up on the IoT platform, i.e. the system design should be such that, for instance, transmission of the next downlink datagram would not occur until the previous downlink datagram had been sent on to the device. This is because, should a transfer be in progress when the radio link completely fails, the entire downlink buffer within the network will be cleared and any datagrams forwarded to the core network by the CDP server could be lost.

If the cloud application requires notification of MT datagram delivery, the UE application must send back an acknowledgement message itself as there is no provision within NB-IoT or the IoT platform being used.

13.6 Sent confirmation

When using the datagram messaging commands (+NMGS, +NMGR), the device can be configured to provide extra URCs for signaling when messages have arrived on the SARA-N2 module or have been acknowledged by the base station/network

Sent confirmation may be requested as follows:

Command	Response	Description
`AT+NSMI=0`	`+NSMI:OK` `OK`	Turns off `+NSMI` URC.
`AT+NSMI=1`	`+NSMI:OK` `OK`	After a datagram has been successfully sent and acknowledged by the network the module will issue the `+NSMI:SENT` URC.

Sent Notifications only acknowledge when the protocol stack has accepted the message on the internal queue, not that it has been acknowledged by the BTS or network.

The state of the send message indications may be queried as follows:

Command	Response	Description

`AT+NSMI?`	`+NSMI=1`
	`OK`
		The default value is 0, in which case no confirmation is given. The command setting is not stored in the module NVM hence, if the parameter was set to 1 and is later read as 0, this means the module has rebooted.

13.7 Receive notification

Notification of receipt of a datagram may be requested as follows:

Command	Response	Description
`AT+NNMI=0`	`+NNMI:OK` `OK`	Turns off the +NNMI URC indications.
`AT+NNMI=1`	`+NNMI:OK` `OK`	When a datagram has been received by the module, the `+NNMI:<length>,<data>` URC will be issued and the datagram will be deleted from the module's internal memory.
`AT+NNMI=2`	`+NNMI:OK` `OK`	When a datagram has been received by the module, the `+NNMI` URC will be issued but the datagram will be retained inside the module. The datagram may then be retrieved from the module using `AT+MGR`.

The state of the new message indications may be queried as follows:

Command	Response	Description

`AT+NNMI?`	`+NNMI=1`
	`OK`
		The default value is 0, in which case the module must be polled with `AT+NMGR` for received messages. This value is not stored in the module NVM hence, if the parameter was set to 1 or 2 and is later read as 0, this means the module has rebooted.

Appendix

A List of acronyms

Abbreviation / Term	Explanation / Definition
AMQP	Advanced Message Queue Protocol
AT	Attention
CDP	Connected Device Platform
DC	Data Channel
DCE	Data Communications Equipment
DL	Downlink
DTE	Data Terminal Equipment
GPRS	General Packet Radio Service
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Station Identity
IP	Internet Protocol
MCS	Message Coding Scheme
MO	Mobile Originated
MT	Mobile Terminated
NB-IoT	Narrow Band Internet of Things
SIM	Subscriber Identity Module
SMS	Short Message Service
TAU	Tracking Area Update
TE	Terminal Equipment
UE	User Equipment
UL	Uplink
URC	Unsolicited Result Code
UUID	Unique User Identification

Related documents

[1] u-blox SARA-N2 series Data Sheet, Docu No UBX-15025564

[2] u-blox SARA-N2 AT Commands Manual, Docu No UBX-16014887

[3] 3GPP TS 36.321 Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification

☞ For regular updates to u-blox documentation and to receive product change notifications, register on our homepage.

Revision history

Revision	Date	Name	Status / Comments
R01	25-Nov-2016	pwar	Initial release
R02	02-Feb-2017	pwar	Emphasis more on UDP. Include protocol states and IP transport diagrams

Contact

For complete contact information visit us at www.u-blox.com

u-blox Offices

North, Central and South America

u-blox America, Inc.

Phone:	+1 703 483 3180
E-mail:	info_us@u-blox.com

Regional Office West Coast:

Phone:	+1 408 573 3640
E-mail:	info_us@u-blox.com

Technical Support:

Phone:	+1 703 483 3185
E-mail:	support_us@u-blox.com

Headquarters
Europe, Middle East, Africa

u-blox AG

Phone:	+41 44 722 74 44
E-mail:	info@u-blox.com
Support:	support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.

Phone:	+65 6734 3811
E-mail:	info_ap@u-blox.com
Support:	support_ap@u-blox.com

Regional Office Australia:

Phone:	+61 2 8448 2016
E-mail:	info_anz@u-blox.com
Support:	support_ap@u-blox.com

Regional Office China (Beijing):

Phone:	+86 10 68 133 545
E-mail:	info_cn@u-blox.com
Support:	support_cn@u-blox.com

Regional Office China (Chongqing):

Phone:	+86 23 6815 1588
E-mail:	info_cn@u-blox.com
Support:	support_cn@u-blox.com

Regional Office China (Shanghai):

Phone:	+86 21 6090 4832
E-mail:	info_cn@u-blox.com
Support:	support_cn@u-blox.com

Regional Office China (Shenzhen):

Phone:	+86 755 8627 1083
E-mail:	info_cn@u-blox.com
Support:	support_cn@u-blox.com

Regional Office India:

Phone:	+91 80 4050 9200
E-mail:	info_in@u-blox.com
Support:	support_in@u-blox.com

Regional Office Japan (Osaka):

Phone:	+81 6 6941 3660
E-mail:	info_jp@u-blox.com
Support:	support_jp@u-blox.com

Regional Office Japan (Tokyo):

Phone:	+81 3 5775 3850
E-mail:	info_jp@u-blox.com
Support:	support_jp@u-blox.com

Regional Office Korea:

Phone:	+82 2 542 0861
E-mail:	info_kr@u-blox.com
Support:	support_kr@u-blox.com

Regional Office Taiwan:

Phone:	+886 2 2657 1090
E-mail:	info_tw@u-blox.com
Support:	support_tw@u-blox.com