

Skywire[®]

Software Developer's Guide

NimbeLink Corp

Updated: August 2018



© NimbeLink Corp. 2018. All rights reserved.

NimbeLink Corp. provides this documentation in support of its products for the internal use of its current and prospective customers. The publication of this document does not create any other right or license in any party to use any content contained in or referred to in this document and any modification or redistribution of this document is not permitted.

While efforts are made to ensure accuracy, typographical and other errors may exist in this document. NimbeLink reserves the right to modify or discontinue its products and to modify this and any other product documentation at any time.

All NimbeLink products are sold subject to its published Terms and Conditions, subject to any separate terms agreed with its customers. No warranty of any type is extended by publication of this documentation, including, but not limited to, implied warranties of merchantability, fitness for a particular purpose and non-infringement.

NimbeLink is a registered trademark, and Skywire is a registered trademark, of NimbeLink Corp. All trademarks, service marks and similar designations referenced in this document are the property of their respective owners.

Table of Contents

Table of Contents	2
Introduction	4
Scope	4
Getting Started	5
Introduction	5
User Manual	5
Application Notes	5
AT Command Manuals	5
Microcontroller Platform	6
Introduction	6
Connections	6
AT Command Data Path Overview	6
Issuing AT Commands	6
Handling AT Command Response	7
Delay Between AT Commands	7
Data Protocol Path	7
Firmware Over The Air (FOTA) Updates	7
SSL/TLS Stacks	8
Hardware Flow Control	8
The Skywire as a State Machine	8
Windows Platforms	9
Introduction	9
Connections	9
Data Protocol Path	9
Firmware Over The Air (FOTA) Updates	10
SSL/TLS Stacks	10
Hardware Flow Control	10
Skywire Monitoring	10
Linux Platform	11
Introduction	11
Connections	11
Data Protocol Path	11
Firmware Over The Air (FOTA) Updates	11

SSL/TLS Stacks	12
Hardware Flow Control	12
Skywire Monitoring	12
Skywire Recovery and Monitoring	13
Introduction	13
Network Registration and Status	13
Modem Recovery	13
Modem Monitoring	13
Skywire Logging	14
Persistent Connections	14
ERROR Results from AT Commands	14

1. Introduction

1.1 Scope

The aim of this document is to serve as a high-level overview of how you can integrate the Skywire cellular modem into your product and application. Specifically, this covers considerations that must be made when developing a device with cellular connectivity on different host controller platforms. Section 2 covers getting started with your Skywire, Section 3 Skywire usage with a microcontroller platform, Section 4 covers Skywire usage in a Windows environment, and Section 5 covers Skywire usage in a Linux environment. Finally, Section 6 covers general usage and monitoring principles to take into consideration when developing with the Skywire.

2. Getting Started

2.1 Introduction

This section provides a path for what to do when you first receive your Skywire modem.

2.2 User Manual

Each Skywire has a User Manual linked in its product page. The User Manual is designed to get customers started with their Skywire for the first time. It also goes through making sure that your Skywire and SIM card (if applicable) is working and registered on the network.

Completing this document is required. Specific application notes are written with the assumption that this document has been read and the processes outlined completed.

2.3 Application Notes

Once you have completed the User Manual to make sure that your Skywire is working properly, NimbeLink has written multiple Application Notes covering the functionality of the Skywire. This includes sending SMS messages, sending HTTP using the built-in TCP stack, and connecting the device to a computer to use as a data pipe.

These application notes outline a proof-of-concept and walkthrough for completing a specific function that the Skywire is capable of. These application notes do not cover every possible configuration option. For that, customers should consult the AT Command Manual.

2.4 AT Command Manuals

Each Skywire has a specific AT Command Manual that should be referenced for descriptions of AT commands, examples, and capabilities. These manuals are usually provided by the module manufacturer, and are the most thorough resource for Skywire capabilities and functionality.

3. Microcontroller Platform

3.1 Introduction

One great feature of the Skywire family of modems is the ability to connect to multiple host devices for use. This section covers using the Skywire with a microcontroller platform.

3.2 Connections

All Skywire modems have a UART path for AT communications, which is the connection path most commonly used for use on a microcontroller platform. Most Skywires also have a USB communications path which can be used by your application as well.

On a microcontroller, you have two options for data communications: direct AT commands using socket dials, or running a data protocol in an RTOS environment.

3.3 AT Command Data Path Overview

The communication standard used for using and configuring the Skywire modem is the AT command standard. Many Skywires include standard commands from the Hayes AT command set and 3GPP standard AT command set. Additionally, many vendors implement their own proprietary AT commands for various functions.

When using a microcontroller, the primary network communication method is Socket Dials. Socket Dials allow data to be sent over TCP or UDP. This enables the use of higher-level protocols such as HTTP or MQTT. Many Skywires have built-in HTTP stacks that can be used. If your Skywire does not have a protocol that you need, you would need to implement an external stack for that protocol. This would be implemented, run, and managed on your microcontroller or in your RTOS environment. NimbeLink does not provide support for implementing external protocol stacks.

Some Skywires have FTP clients as well, enabling FTP downloads.

Please see your Skywire's AT command manual and application notes for more information on available protocols and functionality using AT commands.

3.4 Issuing AT Commands

When issuing AT commands, there are two important rules to adhere to:

1. You must handle the response of an AT command before moving on to the next AT command.
2. You must wait between 50-200 milliseconds (ms) between receiving the response of the last AT command and issuing the next AT command.

3.4.1 Handling AT Command Response

Your application must handle the response to one AT command before moving on to the next AT command. This ensures that you are not issuing AT command too quickly, and also helps your host processor keep in sync with the Skywire. This also alerts the host processor to an error response, which needs to be handled.

Not handling the response can cause AT commands to be issued too quickly, an ERROR response to be missed and not handled, and can get your host processor out of sync, issuing the wrong AT commands.

For more information on handling errors, please see Section 5: Skywire Monitoring and Recovery.

3.4.2 Delay Between AT Commands

Your application must insert a delay between AT commands to let the Skywire process the previous command successfully. This wait time is measured starting at the time you receive the full response to the previous AT command and ending when you send the next AT command. This time is not measured between the times between issuing two AT commands.

Certain manufacturers recommend specific wait times, and those times can be found in their respective AT command manuals. If no specific time is mentioned, NimbeLink recommends 150-200 ms as a wait time.

3.5 Data Protocol Path

Just as you can run a PPP stack on a desktop operating system, you can implement the same protocol into your RTOS environment to send data. PPP is the most commonly used protocol to implement in an RTOS if not using socket dials.

NimbeLink does not provide support for implementing external protocol stacks into RTOS environments. We do, however, have application notes for getting PPP running on a Linux platform located on each Skywire's webpage. Additionally, NimbeLink maintains working PPP scripts on a GitHub page:

<https://github.com/NimbeLink/skywire-ppp-scripts>

Though designed for Linux, these application notes and PPP scripts can serve as a foundation for integrating PPP into your RTOS environment.

3.6 Firmware Over The Air (FOTA) Updates

NimbeLink recommends, and cellular carriers require, having a process in place to update both the firmware on your microcontroller and the firmware on your Skywire modem using FOTA updates. For some carriers, FOTA update for the Skywire modem is required to maintain carrier certifications. Having FOTA update capability for your microcontroller as well allows you to remotely fix bugs and add new features.

Please see your Skywire's page for the FOTA update procedure for your Skywire modem.

3.7 SSL/TLS Stacks

As security becomes more and more important in end-device applications, customers are encouraged to use secure data connection methods. Some Skywires have built-in SSL/TLS stacks, and some do not.

For the modems that have the SSL/TLS stack built-in, please see the associated Application note and AT command manual for more information about supported versions.

If your modem does not have a built-in SSL/TLS stack, or if your application requires a version of TLS that is not supported by your Skywire, you will need to use a stack on your host processor. Examples of this are mbed TLS and wolfSSL.

NimbeLink does not provide support for implementing external SSL/TLS stacks.

3.8 Hardware Flow Control

NimbeLink recommends implementing hardware flow control into your microcontroller application, particularly if you are sending large amounts of data, are sending data very quickly, or both. The Skywire is data circuit-terminating equipment (DCE), and needs to be connected as follows:

- CTS on the Skywire connected to CTS on the host processor
- RTS on the Skywire connected to RTS on the host processor

NimbeLink recommends verifying that flow control is functioning as part of the design if using UART as the primary communication method.

3.9 The Skywire as a State Machine

It can be helpful to interact with and treat the Skywire as a state machine. Your host processor must know what state the Skywire is in, and take action if it is in an unknown state.

For example, if you cannot open a socket, step back and verify that your PDP context is open and you have an IP address. If you don't have an IP address, verify that your APN is correct. If your APN is correct, verify you are registered on the network and have good signal.

For more considerations, please see Section 6: Skywire Monitoring and Recovery.

4. Windows Platforms

4.1 Introduction

The Skywire can also be used with embedded and desktop-versions of Windows. This section covers options for customers looking to use the Skywire modem with Windows.

4.2 Connections

When using the Skywire in Windows, customers will most often use a data path like PPP, NCM, or MBIM to setup a network interface on their Windows device. It is still possible, however, to use the Skywire using a direct AT communications path, as well.

For more information about options using AT commands, please see Sections 3.3 and 3.4.

4.3 Data Protocol Path

Depending on the Skywire that you are using, you may have multiple data paths available to you. PPP is available on Windows by setting up a dial-up modem connection. For Skywires that support PPP, you can use the UART or USB communications path.

Additionally, your Skywire may have a USB data path available as well. The Windows operating system requires a USB driver be installed before the Skywire modem can be recognized by the OS. Each Skywire modem has a different USB driver and the download files are provided in the documentation section for each specific modem.

Some Skywires have native interfaces like NDIS or MBIM available, while others require some setup and management in order to implement.

Finally, different versions of Windows can mean that different data paths on the same Skywire are available. For example, MBIM might be supported in Windows 8 and 10, but not Windows 7.

Using a data path such as PPP or MBIM where Windows handles the device like any other network connection means you can take advantage of the built-in protocol stacks in your device. Additionally, it may be easier to install external stacks (like MQTT) into Windows via a standard software installation procedure compared to trying to integrate it into an RTOS.

NimbeLink does not have any application notes specifically for setting up PPP on Windows. We do, however, have application notes for getting PPP running on a Linux platform located on each Skywire's webpage. Additionally, NimbeLink maintains working PPP scripts on a GitHub page:

<https://github.com/NimbeLink/skywire-ppp-scripts>

Though designed for Linux, these application notes and PPP scripts can serve as a foundation for integrating PPP into your Windows environment.

For Windows communications options, please see your Skywire's page and AT command manual.

4.4 Firmware Over The Air (FOTA) Updates

For more information on FOTA and when it is required, please see Section 3.5.

In addition to FOTA using AT commands, Windows allows you to implement your own FOTA update process using the normal Windows update utilities provided by Microsoft. For example, you can download the update file and use the command line tool to update your Skywire. This could be automated within your Windows device. See the Firmware update application notes specific for your selected Skywire modem for more details.

4.5 SSL/TLS Stacks

One advantage of using the Skywire as a network interface is the ability to use the built-in security stacks in Windows. This saves development time integrating external stacks.

4.6 Hardware Flow Control

If you plan on using the Skywire's UART communications path with your Windows device, NimbeLink recommends implementing hardware flow control. For more information on flow control, please see Section 3.8.

If you are using the USB data path, you do not need to setup hardware flow control as it is included in the USB standard.

4.7 Skywire Monitoring

Any sort of connection and network monitoring will need to be built in to your end device to maintain a reliable connection. Please see Section 6 for considerations for monitoring the Skywire.

5. Linux Platform

5.1 Introduction

The most commonly-used and well-supported full operating system for Skywire modem is Linux, both in embedded and desktop flavors. This section covers using the Skywire in a Linux environment.

5.2 Connections

Due to the ability to create custom Linux distributions, and the widespread availability of pre-built distributions, many customer use the UART data path, USB data paths, or both with Linux.

Most Skywires can use PPP, and many have USB data paths available as well.

For more information on the data paths available to you, please see your Skywire's webpage.

Finally, you can use AT commands to control the Skywire. For more information about options using AT commands, please see Sections 3.3 and 3.4.

5.3 Data Protocol Path

You may have multiple data paths available to you depending on your Skywire. PPP is available on Linux, and most Skywire supporting PPP can use either UART or USB.

Additionally, your Skywire may have a USB data path available as well. Some Skywires have native interfaces like CDC-ECM available, while others require some setup and management in order to implement.

Using a data path such as PPP or a USB data path allows Linux to handle the device like any other network connection. This means you can take advantage of the built-in protocol stacks in your device. Additionally, it may be easier to install external stacks (like MQTT) into Linux via a standard software installation procedure compared to trying to integrate it into an RTOS.

NimbeLink has written guides for most Skywires for using PPP on Linux. Additionally, we host example PPP scripts as a starting point for your application:

<https://github.com/NimbeLink/skywire-ppp-scripts>

Additionally, we have written guides using the USB data path for most Skywires as well.

For Linux data paths, please see your Skywire's AT command manual and application notes.

5.4 Firmware Over The Air (FOTA) Updates

For more information on FOTA and when it is required, please see Section 3.5.

In addition to FOTA using AT commands, Linux allows you to implement your own FOTA update process using the Linux update utilities the NimbeLink provides. For example, you can download the update file and use the command line tool to update your Skywire. This could be automated within your Linux device.

5.5 SSL/TLS Stacks

One advantage of using the Skywire as a network interface is the ability to use the built-in security stacks in Linux. This saves development time integrating external stacks.

5.6 Hardware Flow Control

If you plan on using the Skywire's UART communications path with your Linux device, NimbeLink recommends implementing hardware flow control. For more information on flow control, please see Section 3.8.

If you are using the USB data path, you do not need to setup hardware flow control as it is included in the USB standard.

5.7 Skywire Monitoring

Any sort of connection and network monitoring will need to be built in to your end device to maintain a reliable connection. Please see Section 6 for considerations for monitoring the Skywire.

6. Skywire Recovery and Monitoring

6.1 Introduction

When designing a robust and reliable system using a cellular modem, regardless of the host processor being used, additional software on the host processor must be written for modem management. This section covers some characteristics of the management software to be taken into consideration. Not everything mentioned in this section will be required for each customer's application, but most of these ideas are needed for consistent performance.

6.2 Network Registration and Status

The first thing to check and monitor is that the modem has cellular signal and a network connection. Regardless of how a customer plans to integrate the Skywire into their system, an application will need to be registered on the network and have cellular signal in order to function. If these basic characteristics are not met, the cellular modem will not function.

NimbeLink recommends that checking network status and cellular signal strength should be integrated into your recovery software.

6.3 Modem Recovery

There may be times where you need to recover the modem from an unknown state into a known state, depending on the state you are in. For example, if you are unable to open a socket, go back and check that the modem has an IP address. If it doesn't have an IP address, verify that the context is activated and that it is registered on the network.

You can also issue AT commands to do a hardware reset of the modem. Please see the respective AT command manual for more information on that command.

Finally, each Skywire has a reset pin. Usage of this reset pin is recommended as a last resort as it "ungracefully" removes the modem from the network, which can cause damage to the modem and could potentially corrupt the flash contents of the Skywire. Please see the datasheet of your respective Skywire for using the reset pin.

6.4 Modem Monitoring

If you are using the Skywire as a data connection for a host operating system (for example, as a PPP or CDC-ECM connection for a Linux operating system), you will need to use a network manager to monitor the connection and take action if needed. This can be a watchdog timer that you create, or you can modify an existing software project for your needs. This manager will need to periodically check or watch the network connection, and take action if it goes down to bring it back up.

6.5 Skywire Logging

NimbeLink recommends logging output from the Skywire to assist with debugging efforts and general monitoring. This can either mean a constant log of AT command outputs stored on the host processor or sent to a centralized logging system, or a set of AT commands that gets run to check basic functionality when things are not working as expected.

Please note that NimbeLink's support will be limited in assisting customers who have devices in the field that are no longer responding and are not sending in logs periodically. Additionally, it will be difficult for the designers of the application to perform debugging without proper logging and reporting.

6.6 Persistent Connections

If a persistent connection is required by your application, you must implement monitoring and "keepalive" software in order to make sure you have a persistent connection. Carriers will remove idle devices from their networks in order to prioritize active devices, so your host processor must check and re-establish the connection if this happens.

6.7 ERROR Results from AT Commands

There may be times when an AT command returns a response of ERROR. Your host processor must be able to handle these errors. NimbeLink recommends retrying the command a certain number of times, and stepping back to verify functionality if the ERROR persists.