WINC1500 Software

Release Notes

VERSION : **19.6.1**

DATE : **JUL 09, 2018**

Abstract

This document presents an overview of the WINC15x0 firmware release version 19.6.1, and corresponding driver.

1 Introduction

This document describes the WINC15x0 version 19.6.1 release package.

The release package contains all the necessary components (binaries and tools) required to make use of the latest features including tools, and firmware binaries.

1.1 Highlights of the release

1.1.1 Features

- **WLAN**
  - New Enterprise Security Methods:
    - EAP-PEAP/MSCHAPv2 (PEAPv0 and PEAPv1)
    - EAP-TLS
    - EAP-PEAP/TLS (PEAPv0 and PEAPv1)
  - Multiple Gain Table Support
  - Simple Roaming
  - New Connection API allowing more functionality
- Network Stack
  - Customisable NTP servers
  - TCP Keepalive timeout Socket Option
- OTA
  - Host File Download (WINC1510, 8Mbit Flash only)
- Miscellaneous
  - Encrypted Credential Storage
  - Flash Tool Verification
  - Built in ATE Mode Support

1.1.2 Notable changes

- The DBG UART baud rate is now set to 460800
- Provisioning page changes
- Subnet mask/Default Router/DNS in AP mode

1.2 Firmware readiness

Microchip Technology Inc. considers version 19.6.1 firmware to be suitable for production release

2 Release summary

2.1 Auditing information

Master Development Ticket	:	https://jira.microchip.com:8443/projects/W1500/versions/36824
Release Repository	:	Wifi_M2M
Source Branch	:	/branches/rel_1500_19.6.1
Subversion Revision	:	**r16916**

2.2 Version information

WINC Firmware version	:	19.6.1
Host Driver version	:	19.6.1
Minimum driver version	:	19.3.0

Please note that the SVN revision advertised in the firmware serial trace will be **16761**.

```
(10)NMI M2M SW  VER 19.6.1 REV 16761
(10)NMI MIN DRV VER 19.3.0
(10)FW URL branches/rel_1500_19.6.1
(10)Built May 23 2018   14:39:16
```

2.3 Released components

The release contains documentation, sources and binaries.

2.3.1 Documentation overview

The Application manuals, Release notes and Software API guides can be found in the `doc/` folder of the release package.

Release Notes:

This document

Software APIs:

WINC1500_IoT_SW_APIs.chm

2.3.2 Binaries and programming scripts

The main WINC15x0 firmware binary is located in the `firmware` directory and named `m2m_aio_3a0.bin`. This can be flashed to a WINC device using, for example, a serial bridge application available from ASF.

An OTA image is provided in the `ota_firmware` directory named `m2m_ota_3a0.bin`.

2.3.3 Sources

Source code for the host driver can be found under the `src/host_drv` directory.

Source code for the tools can be found under the `src/Tools` directory.

2.4 Release Comparison

Features in 19.5.4	Changes in 19.6.1
Wi-Fi STA	

• IEEE 802.11 b/g/n.
• OPEN, WEP security.
• WPA Personal Security (WPA/WPA2).
• WPA Enterprise Security (WPA/WPA2) supporting EAP-TTLSv0/MSCHAPv2 authentication with RADIUS server.
	• WPA/WPA2 Enterprise new methods:
	  ○ EAP-PEAPv0/MSCHAPv2
	  ○ EAP-PEAPv1/MSCHAPv2
	  ○ EAP-PEAPv0/TLS
	  ○ EAP-PEAPv1/TLS
	  ○ EAP-TLS
	• WPA/WPA2 Enterprise other new features:
	  ○ Phase 1 TLS session caching
	  ○ Option to specify domain
	  ○ Option to send actual identity in phase 1
	• Simple Roaming support.
	• Improved connection API, allowing connection via BSSID as well as SSID.
	• Option to encrypt connection credentials that are stored in WINC15x0 flash.

Wi-Fi Hotspot	
• Only ONE associated station is supported. After a connection is established with a station, further connections are rejected. • OPEN and WEP, WPA2 security modes. • The device cannot work as a station in this mode (STA/AP Concurrency is not supported).	No change
Wi-Fi Direct	
• Wi-Fi direct client is not supported.	No change
WPS	
The WINC15x0 supports the WPS protocol v2.0 for PBC (Push button configuration) and PIN methods.	No change
TCP/IP Stack	
The WINC15x0 has a TCP/IP Stack running in firmware side. It supports TCP and UDP full socket operations (client/server). The maximum number of supported sockets is currently configured to 11 divided as: • 7 TCP sockets (client or server). • 4 UDP sockets (client or server).	No change

Features in 19.5.4	Changes in 19.6.1
Transport Layer Security	
• Support TLS v1.2. • Client and server modes. • Mutual authentication. • Custom scheme for X509 certificate revocation. • X509 certificate support including SHA1, SHA256, SHA384 and SHA512. • Integration with ATECC508 (adds support for EC-DSA and ECDHE). • Supported cipher suites are: TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_DHE_RSA_WITH_AES_128_CBC_SHA TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 TLS_DHE_RSA_WITH_AES_256_CBC_SHA TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (requires ATECC508) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (requires ATECC508) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (requires ATECC508)	No change
Networking Protocols	
DHCPv4 (client/server) DNS Resolver IGMPv1, v2. SNTP	• SNTP servers are fully customisable

Features in 19.5.4	Changes in 19.6.1
Power saving Modes	
• M2M_PS_MANUAL • M2M_PS_AUTOMATIC • M2M_PS_H_AUTOMATIC • M2M_PS_DEEP_AUTOMATIC	No change
Device Over-The-Air (OTA) upgrade	
• Built-in OTA upgrade available. • Backwards compatible as far as 19.4.4, with the exception of: - Wi-Fi Direct (removed in 19.5.3) - Monitor mode (removed in 19.5.2)	No change
Wi-Fi credentials provisioning via built-in HTTP server	
Built-in HTTP/HTTPS (TLS server mode) provisioning using AP mode (Open, WEP or WPA2 secured).	• Improved provisioning user experience • Default gateway and subnet mask can now be customised when in AP mode.
Ethernet Mode (TCP/IP Bypass)	
Allow WINC1500 to in WLAN MAC only mode and let the host to send/receive Ethernet frames.	No change
ATE Test Mode	
Embedded ATE test mode for production line testing driven from the host MCU.	No change.
Miscellaneous features	
	• Addition of host file download capability, allowing the host MCU to download and retrieve files from the WINC15x0 flash.

3 Test Information

Please refer to ticket W1500-393 and the test report issued with this release for full details.

Testing was performed against the release candidate 19.6.1 against the following configuration(s):

H/W Version : WINC1510 XPLAINED PRO module

Host MCU : ATSAMD21-XPLAINED PRO

For testing involving elliptic curve cryptography, the host MCU also communicated with an ECC508A chip on a CRYPTOAUTH XPLAINED PRO board.

The following testing was performed in both open air and shielded automated environments:

1. **General 802.11 WiFi functionality**
    1. Provisioning
        - HTTP
        - HTTPS
    2. Scanning
        - Active scan
        - Passive scan
    3. Station Mode
        - WPA/WPA2/WEP/Open security
        - Throughput testing
    4. AP Mode
        - WPA2/WEP/Open security
        - Throughput testing
    5. WPS
        - PIN
        - Push Button
    6. Roaming
        - Channel change and link loss
        - Traversing subnets
    7. 802.1x Enterprise (with WPA/WPA2 and TLS caching on/off)
        - EAP-TTLS-MSCHAPv2
        - EAP-TLS
        - EAP-PEAPv0-TLS
        - EAP-PEAPv1-TLS
        - EAP-PEAPv0-MSCHAPv2
        - EAP-PEAPv1-MSCHAPv2
    8. Bypass mode (STA)
        - WPA/WPA2/WEP/Open security
        - Enterprise (as per above)

2. **Over-The-Air (OTA) update functionality and robustness**
    - Upgrade/Rollback
    - HTTP/HTTPS including ECC and client authentication
    - Local/Remote (AWS) server
    - Destructive scenarios, including client and server disconnect before completion

3. **Backwards compatibility**
    - 19.4.4 toolchain and driver; firmware updated to 19.6.1
    - 19.5.4 toolchain and driver; firmware updated to 19.6.1
    - 19.4.4 toolchain; driver and firmware updated to 19.6.1

4. **TLS functionality**
    1. RSA cipher-suites:
        - TLS_RSA_WITH_AES_128_CBC_SHA
        - TLS_RSA_WITH_AES_128_CBC_SHA256
        - TLS_RSA_WITH_AES_128_GCM_SHA256
        - TLS_DHE_RSA_WITH_AES_128_CBC_SHA
        - TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
        - TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

        Testing uses a 1024-bit server certificate, with a chain of 7 certificates of varying key lengths (1024,2048 and 4096 bit) leading to a 2048-bit root certificate.
    2. ECDSA cipher-suites:
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

        Testing uses a NIST standard ECC P256 prime curve server certificate with two chains, one leading back to an ECDSA root certificate and the other leading to an RSA root certificate.
    3. Client authentication (both RSA and ECDSA client certificates)

5. **Performance under interference**
    1. Most tests are executed in shielded environments under different controlled levels and types of interference, including:
        - Co-channel (interferer on same channel)
        - Adjacent (interferer one channel away)
        - 11b (interferer is sending very large 11b frames)
        - Heavy (high level of interference)
        - Extreme (Interferer is near to saturation)

6. **TCP/IP stack robustness testing**
    1. Using an internal implementation of IPerf to verify UDP and TCP throughput scenarios
    2. Verification of multi socket functionality
    3. HTTP POST/GET

7. **Host File Download**
    1. Remote download and verification

8. **Wi-Fi AP interoperability**

9. **Manual Wi-Fi browser interoperability**

4 Known Issues

ID	Description
W1500-63	Occasionally WINC15x0 fails to receive an individual UDP broadcast frame when in M2M_PS_DEEP_AUTOMATIC powersave mode. Recommended workaround: Use M2M_NO_PS powersave mode if reliability is preferred for UDP broadcast frames. Otherwise ensure the overlying protocol can handle the odd missing frame.
W1500-108	The WINC15x0 cannot handle two simultaneous TLS handshakes, due to memory constraints. Recommended workaround: When attempting to open two secure sockets in STA mode, the application should wait to be notified of the first one completing (succeeding or failing) before attempting the second one.
W1500-177	Under high interference and high data throughput (TCP/UDP), the WINC15x0 occasionally runs out of memory for receiving data and does not recover. This occurred 4 times during 9 hours of high interference high throughput rx/bidirectional testing. Recommended workaround: Close all sockets then retry the data transfer.
W1500-325	1% of Enterprise conversations fail due to the WINC15x0 not sending an EAP response. The response is prepared and ready to send but does not appear on the air. After 10 seconds the firmware times-out the connection attempt and the application is notified of the failure to connect. Recommended workaround: Configure the authentication server to retry EAP requests (with interval < 10 seconds). The application should retry the connection request when it is notified of the failure.
W1500-344	Using the `m2m_wifi_set_tx_power()` API stops the WINC15x0 from transmitting. Recommended workaround: Avoid using the `m2m_wifi_set_tx_power()` API.

W1500-369	When connected to certain access points, the WINC15x0 sometimes fails to roam when the access point changes channel. The issue is seen with these access points: Linksys E2500, Linksys E4200, Linksys 6500. The failures to roam are due to two issues: 1. Sometimes the access point takes a long time to start sending beacons or probe responses on the new channel, so it is not discoverable. 2. Sometimes the access point does not initiate the 4-way handshake (for WPA/WPA2 PSK reconnections). Recommended workaround: On reception of M2M_WIFI_DISCONNECTED event, the application should attempt to discover the access point using `m2m_wifi_request_scan()` API.
W1500-370	When provisioning the WINC15x0 using a mobile phone, 5% of provisioning attempts cause an error message “Request Failed” to pop up on the phone, even though the provisioning has succeeded. Recommended workaround: Ignore the “Request Failed” message.
W1500-381	When connecting to a TL-WR841N router, data transfer is sometimes unavailable until several seconds after DHCP. Occasionally the data-plane is never established. Recommended workaround: If DHCP completes but data transfer fails, disconnect and reconnect to the router.
W1500-387	If an AP uses an 802.11 ACK policy of “No Ack”, then the WINC15x0 sometimes fails to receive 802.11b frames. Recommended workaround: Avoid using an ACK policy of “No Ack”. If “No Ack” is used, ensure frames are sent at 802.11g or higher rates.
W1500-397	70% of Enterprise connection requests fail with a TP Link Archer D2 access point (TPLink-AC750-D2). The access point does not forward the initial EAP Identity Response to the authentication server. The issue is bypassed by PMKSA caching (WPA2 only), so reconnection attempts will succeed. Recommended workaround: The application should retry the connection request when it is notified of the failure.

5 New Features

5.1 New Enterprise Security Methods

There has been a significant increase in support for Wi-Fi connection in STA mode with WPA/WPA2 Enterprise.

This new feature section describes the support in 19.6.1, even though some of the support was present in previous releases. Please see section 2.4 for a summary of changes from previous releases.

5.1.1 Authentication Methods

Here is a list of authentication methods supported by WINC15x0.

- EAP-TTLSv0/MSCHAPv2
- EAP-PEAPv0/MSCHAPv2
- EAP-PEAPv1/MSCHAPv2
- EAP-PEAPv0/TLS
- EAP-PEAPv1/TLS
- EAP-TLS

Enterprise authentication comprises server authentication and client authentication.

Most authentication methods are two-phase methods. Server authentication is done during phase 1 (EAP-TTLS or EAP-PEAP) and client authentication is done during phase 2 (MSCHAPv2 or TLS). Phase 2 is encrypted by a secure channel (‘tunnel’) which is set up during phase 1.

EAP-TLS is a single-phase method and includes both server authentication and client authentication.

5.1.1.1 Server Authentication

Server authentication is always done via TLS; the server must provide a certificate chain leading to a root certificate that is trusted by the client.

In the case of the WINC15x0 client, trusted root certificates are stored in flash. These are typically written by a tool during production. It is recommended that a good range of trusted root certificates is included.

5.1.1.2 Client Authentication

Client authentication may be done by any method that is supported by both server and client; the credentials used depend on the method.

In the case of the WINC15x0 client, either MSCHAPv2 or TLS may be supported - the application provides credentials for one or other method as part of the connection request API (see 5.4).

5.1.2 Connection Caching

Caching significantly reduces reconnection time, which is especially important during roaming (see section 5.3).

Here is a list of the caching methods supported by WINC15x0.

- PMKSA (access point must be WPA2)
- TLS session resume for EAP-TTLSv0, EAP-PEAPv1 and EAP-TLS.

5.1.2.1 PMKSA Caching

PMKSA caching allows the Enterprise conversation to be skipped entirely when reconnecting to a WPA2 access point. Up to 4 WPA2 access points can be stored in the WINC15x0 PMKSA cache (after which each new entry replaces the oldest entry).

It is generally supported and enabled by WPA2 access points and it is not available for WPA.

5.1.2.2 TLS Session Resume

TLS session resume allows a much shorter Enterprise conversation when connecting to access points that share the same authentication server. Up to 4 entries can be stored in the WINC15x0 TLS session cache.

TLS session resume requires the authentication server to be configured for TLS session caching. For a hostapd server, this is done by adding "`tls_session_lifetime=3600`" to the *hostapd.conf* file.

5.1.3 Credential Handling

5.1.3.1 Domain

A domain can be provided, if required for routing to the appropriate authentication server.

WINC15x0 supports any domain format, including: "user@domain" and "domain\user". The application must include any separators ('@' or '\') when providing the domain in the connect API. The total length of user name and domain, including separators, is limited to a maximum of 132 characters.

5.1.3.2 Anonymous Identity

In two-phase authentication, it is recommended that the client use an anonymous identity during phase 1 and its actual identity during phase 2. In single-phase authentication (EAP-TLS), the client must use its actual identity in phase 1. Additionally, some authentication servers may require the actual identity in phase 1 even for two-phase authentication.

WINC15x0 connection APIs provide an option to specify whether "anonymous" or the actual identity is used during phase 1.

5.1.3.3 TLS Client Credentials

When TLS is used for client authentication, the client sends the authentication server a certificate. The certificate must have been signed by the server, and the client must possess the private key corresponding to the certificate.

WINC15x0 must use RSA for client authentication and the key modulus must not be longer than 256 bytes. The application must provide the certificate, private key modulus, and private key exponent in the connect API. The length of the certificate is limited to a maximum of 1584 bytes.

5.1.3.4 Storing Credentials in WINC15x0 Flash

The credentials can optionally be stored in WINC15x0 flash, either encrypted or unencrypted. This allows the application to request a connection via m2m_wifi_default_connect. For more details and security considerations please see section 5.7.

5.1.4 APIs

Here are the APIs that are available to the application for requesting a connection to an Enterprise network:

- `m2m_wifi_connect_1x_mschap2` for connecting to an Enterprise network using MSCHAPv2 credentials. The full authentication method (EAP-TTLSv0/MSCHAPv2, EAP-PEAPv0/MSCHAPv2 or EAP-PEAPv1/MSCHAPv2) will depend on the configuration of the authentication server.
- `m2m_wifi_connect_1x_tls` for connecting to an Enterprise network using TLS client credentials. The full authentication method (EAP-TLS, EAP-PEAPv0/TLS or EAP-PEAPv1/TLS) will depend on the configuration of the authentication server.

- `m2m_wifi_default_connect` for reconnecting to the last connected Enterprise network (assuming a previous connection request used the option to store the credentials in WINC15x0 flash).

Please refer to *WINC1500_IoT_SW_APIs.chm* for full details of these APIs and their parameter types.

5.1.5 Implementation Detail

The WINC15x0 implementation follows the authentication method specifications given in Table 1.

For EAP-PEAPv0 and EAP-PEAPv1, there are some intentional deviations from the specifications in order to improve interoperability with existing server implementations.

Table 1 - Authentication method specifications

Method	EAP type	RFC (or draft)
EAP-TTLSv0	21	rfc5281
EAP-PEAPv0	25	draft-kamath-pppext-peapv0-00
EAP-PEAPv1	25	draft-josefsson-pppext-eap-tls-eap-05
EAP-TLS	13	rfc5216
EAP-MSCHAPv2	26	draft-kamath-pppext-eap-mschapv2-02
TTLSv0/MSCHAPv2	N/A	rfc5281

5.1.5.1 EAP-PEAPv1 Deviations

The WINC15x0 implementation of EAP-PEAPv1 deviates from draft-josefsson-pppext-eap-tls-eap-05 in the following ways:

- Keying material is generated using label “client EAP encryption” (instead of “client PEAP encryption”).
- An EAP-PEAPv1 ACK is sent in response to an encrypted EAP-Success (instead of sending no response).
- The EAP extensions method is handled (as defined for EAP-PEAPv0 in draft-kamath-pppext-peapv0-00).

5.1.5.2 EAP-PEAPv0 Deviations

The WINC15x0 implementation of EAP-PEAPv0 deviates from draft-kamath-pppext-peapv0-00 in the following ways:

- Keying material is generated using label “client EAP encryption” (instead of “client PEAP encryption”).

5.1.6 Test Summary and Interoperability

The WINC15x0 implementation of WPA/WPA2 Enterprise has been tested extensively against a configurable hostapd 2.6 server. Please refer to section 3 and accompanying test report for details.

There has not been extensive testing of interoperability with different server implementations; besides hostapd 2.6, some limited testing has been done against a Cisco server (Aironet 2600 series) and a Ubiquiti server.

There is information online about some differences between different server implementations. Based on this, it is expected that the WINC15x0 client implementation would be interoperable with a wide range of servers, with the exceptions of “Radiator” PEAPv1 and “Lucent NavisRadius” PEAPv1.

5.2 Multiple Gain Table Support

5.2.1 Overview

There are regulations regarding the maximum transmit power of a Wi-Fi device according to the regulatory Wi-Fi region.

An improper combination of transmit power level and antenna gain can result in Equivalent Isotropic Radiated Power (EIRP) that exceeds the amount allowed per regulatory domain. In some situations, the channel selection or country code affects the transmit power level. The FCC and CE regulatory authorities govern the maximum transmit power of Wi-Fi products for a specific Wi-Fi region.

In short, the maximum transmit power is limited according to regulatory Wi-Fi region. The gain table is the way through which we configure the transmission power in WINC devices. Prior to 19.6.0 only one gain table was supported in WINC15x0, and as a result the WINC can only operate in one regulatory region without requiring different flash contents.

If the WINC15x0 device must operate in other regulatory regions as well (without re-flashing the firmware/driver), it must support multiple gain tables and a provision to select a specific gain table (in turn specific transmission power) on the fly.

In the 19.6.0 release, a maximum of 4 gain tables is supported, allowing the same flash image to be used in up to four Wi-Fi regulatory regions. The application developer can select the required table according to the region that the device is determined to be within.

5.2.2 Usage

5.2.2.1 Writing the gain table to the WINC15x0

The gain builder application will take multiple .csv files (up to a maximum of 4) and perform the necessary maths operations on them to calculate the gain values and write them to the flash:

```
gain_builder [-table <no_of_tables> <img_path1> <img_path2> <img_path3> <img_path4>] [-index <gain_table_index>][-no_wait] [-port]
```

The *img_path** parameters specify the separate tables, and the *index* parameter specifies the default table to use on power up.

5.2.2.2 Selecting a specific table

Setting the specific gain table index is achieved through a new host API call `m2m_wifi_set_gain_table_idx()`. For more details refer to *WINC1500_IoT_SW_APIs.chm.*

5.3 Simple Roaming

5.3.1 Overview

802.11 roaming is a feature in which the WLAN STA moves around inside an ESS area formed by multiple Access Points implementing individual BSS's. The act of roaming is to automatically connect to another AP inside the ESS when the existing AP connection is broken.

An AP forming a BSS inside the ESS shares the same SSID, and normally shares the same IP layer subnet as other BSS's. Therefore, roaming enables a station to change its Access Point while remaining connected to the IP layer network.

In 19.6.1, the WINC15x0 roam will occur on link-loss detection with the existing AP, which is determined by tracking beacons and sending NULL frame keep-alive packets.

ISO/OSI Layer 2 roaming occurs when the WINC15x0 roams from one AP to another AP, both of which are inside the same IP subnet. Layer 3 roaming occurs when the WINC15x0 roams from one AP to another AP which are in different subnets, whereby the WINC15x0 will attempt to obtain a new IP address within the new subnet via DHCP. As a result of layer 3 roaming, any existing network connections will be broken, and if is the job of upper layer protocols to handle this IP address change if a continuous connection is required in layers 4 and above.

Below are the steps performed by the WINC15x0 during roaming, once link-loss has been detected with the existing AP:

1. A precautionary de-authentication frame is sent to the old AP
2. Scanning is performed to determine if there is an AP within the same ESS as the previous AP in the vicinity.
3. If an AP is found, authentication and re-association messages are exchanged with the new AP, followed by a normal 4-way security handshake in the case of WPA/WPA2, or an EAPOL exchange in the case of 802.1x Enterprise security.
4. A DHCP request is sent to the new AP to attempt to retain the same IP address. A notification event is sent to the host MCU of type M2M_WIFI_RESP_CON_STATE_CHANGE with the state of M2M_WIFI_ROAMED. Additionally, an M2M_WIFI_REQ_DHCP_CONF event conveying either the same or a new IP address is sent to the host MCU.
5. If there is any problem with the connection, or DHCP fails, then a de-authentication message is sent to the AP, and an M2M_WIFI_RESP_CON_STATE_CHANGED event is sent to the host MCU with the state set as M2M_WIFI_DISCONNECTED.

5.3.2 Usage

5.3.3 Enabling roaming

The API call `m2m_wifi_enable_roaming()` sets the WINC15x0 to detect link-loss more aggressively, and once detected, to perform the roaming steps specified in 5.3.1. For full details, see *WINC1500_IoT_SW_APIs.chm.*

The `bEnableDhcp` parameter enables control of whether or not a DHCP request is sent after roaming to a new AP.

5.3.4 Disabling roaming

The API call `m2m_wifi_disable_roaming()` is used to disable roaming.

5.4 New Connect APIs

This section refers to driver changes only – if 19.6.1 firmware is used with an older driver, this section can be ignored.

New APIs have been added in the 19.6.1 driver for requesting Wi-Fi connection in STA mode.

The legacy APIs (`m2m_wifi_connect` and `m2m_wifi_connect_sc`) are still available as wrappers for the new APIs. Functionally their behavior is unchanged from previously released drivers.

All new connect APIs enable connection to a particular AP by specifying its BSSID as well as the SSID.

It is recommended that if roaming is enabled, a connection is made only to an SSID, without specifying BSSID.

5.4.1 New APIs

Here is a list of the APIs which have been added in 19.6.1:

- `m2m_wifi_connect_open`
- `m2m_wifi_connect_wep`
- `m2m_wifi_connect_psk`
- `m2m_wifi_connect_1x_mschap2`
- `m2m_wifi_connect_1x_tls`

Please refer to *WINC1500_IoT_SW_APIs.chm* for full details of these APIs and their parameter types.

5.4.2 Increased functionality

Here is a summary of the additional functionality provided by these new APIs, compared to the legacy APIs:

- The application may specify a BSSID (in addition to SSID), to restrict connection to a particular access point.
- The application may request that the connection credentials are left unencrypted when stored in WINC15x0 flash (the default behavior of 19.6.1 firmware is to encrypt credentials. See section 5.7 for further information).
- A greater range of credentials may be provided by the application when connecting to an Enterprise network:
    - Domains may be provided.
    - Longer user names may be provided (up to 132 characters).
    - Longer MSCHAPv2 passwords may be provided (up to 256 characters).
    - TLS credentials may be provided.
    - There is an option to send the actual identity (instead of “anonymous”) during phase 1 of an Enterprise authentication.

5.5 Fully Customisable NTP Servers

5.5.1 Overview

In releases prior to 19.6.1, the WINC SNTP client would first try contacting `time.nist.gov`, if that failed it would then try `pool.ntp.org`. These server names were hardcoded and could not be changed.

As of 19.6.1 the NTP server(s) that the WINCs built in SNTP client uses can now be customised using the `m2m_wifi_configure_sntp()` API. The new configuration options allow use of a single NTP server or a pool of servers, and use of an NTP server supplied by DHCP (option 42 - https://www.ietf.org/rfc/rfc2132.txt sec 8.3).

The default setting for the WINC SNTP client is to initially use the NTP server provided by DHCP, if the DHCP server has not provided a server IP or the provided server fails then `time.nist.gov` will be used.

Providing a custom NTP server via the API will overwrite the default `time.nist.gov` server. The custom NTP server can either be a single IP (for example: 178.79.162.34) or a single domain name (for example: `pool.ntp.org`). To use the server pool feature the first character of the provided server name must be an asterisk (for example: `*.pool.ntp.org`). The asterisk will then be replaced with an incrementing value from 0 to 3 each time a server fails. The API can also enable or disable use of the NTP server provided by DHCP.

An NTP update is initiated after successfully connecting to an AP and obtaining an IP via DHCP. The SNTP client will re-sync at most once per day, triggered by a DHCP renew. For example, if the renewal period is 1 hour, then the 48th renewal will trigger an update (the DHCP client renews every half the specified renewal period). If the renewal period is 4 days then the time will be re-synced every 2 days.

Each NTP server is tried 5 times with a timeout of 5 seconds before failing and moving to the next server. If all NTP servers fail then the client will wait for 2 minutes before going through the configured servers again.

See "*Figure 1 - WINC SNTP states*" for a diagram of how the WINC decides which server to use. Refer to *WINC1500_IoT_SW_APIs.chm* for details of how to use the `m2m_wifi_configure_sntp()` *API.*

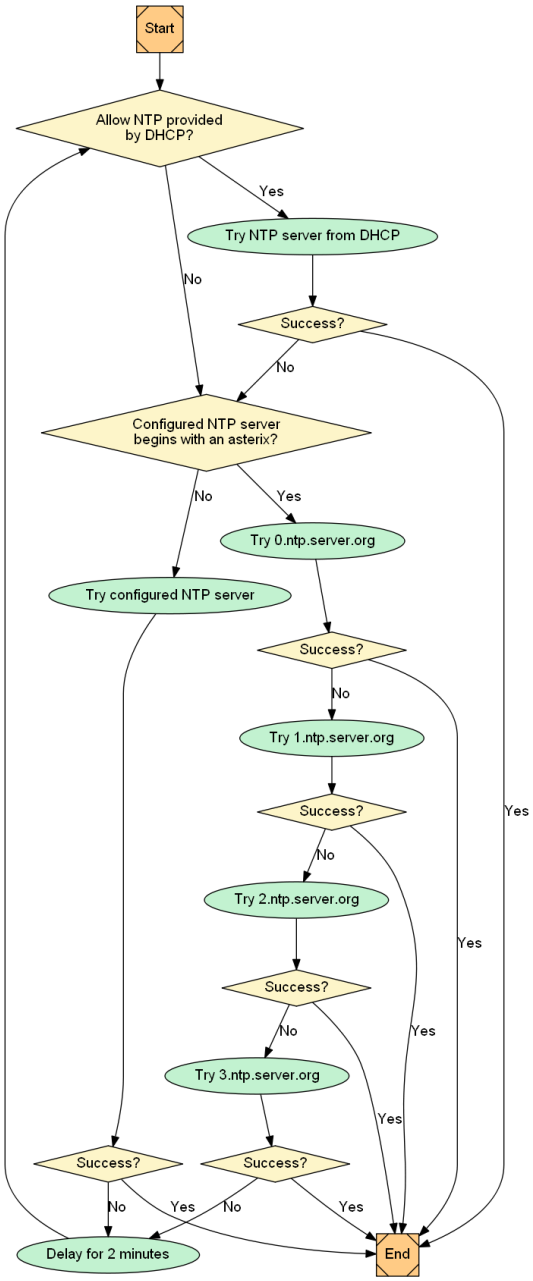

Figure 1 - WINC SNTP states

5.6 TCP Keepalive timeout socket option

5.6.1 Overview

TCP keepalive socket options can now be configured via the `setsockopt()` API. These socket options allow customisation of the keepalive time, interval and probe count for each TCP connection.

Table 2 - TCP Keepalive socket options

Option	Description	Default Setting
SO_KEEPALIVE	Enable or disable keepalive for a TCP socket	Enabled
TCP_KEEPIDLE	Duration between two keepalive transmissions in idle condition	60 seconds
TCP_KEEPINTV	Duration between two successive keepalive retransmissions, if acknowledgement to the previous keepalive transmission is not received	0.5 seconds
TCP_KEEPCNT	Number of retransmissions to be carried out before declaring that the remote end is not available	20

For API implementation details please refer to *WINC1500_IoT_SW_APIs.chm.*

5.6.2 Examples

Table 3 - TCP keepalive examples

Call	Description

`int32 val = 200;`
`setsockopt(sock, 0, TCP_KEEPIDLE, &val, sizeof(val));`
`val = 10;`
`setsockopt(sock, 0, TCP_KEEPINTV, &val, sizeof(val));`
`val = 5;`
`setsockopt(sock, 0, TCP_KEEPCNT, &val, sizeof(val));`
	Set the keepalive idle duration to 100 seconds, resend interval to 5 seconds and retransmit count to 5.

`int32 val = 0;` `setsockopt(sock, 0, SO_KEEPALIVE, &val, sizeof(val));`	Disable TCP keepalive for a socket

5.7 MCU OTA Support

5.7.1 Overview

Host File Download is a feature introduced in software version 19.6.1 and it is specific for WINC1510, since the device has 8Mb of Flash. The feature allows for the application in the Host MCU to download a file and store it in the WINC's Flash. This opens the possibility not only to perform Host MCU OTA updates, but also to access information stored in files downloaded from a remote location. Supports connections via http/https links and it introduces APIs to Download a File, Read the File and Erase the File.

When performing MCU OTA updates, there is no enforced file format, so the developer has flexibility when choosing a strategy to perform integrity check validation on the received file. The WINC does not perform any integrity check on the downloaded file and therefore, it is recommended that the Application do it instead.

The feature is designed for single file support and allows for a maximum size of 508KB. The driver will protect against invalid access to the file stored in the WINC's Flash by using file handlers to identify each file. If a new download starts or if the file is erased, access to the File partition will be denied. Also, the Application can request an explicit erase to delete the file from the WINC's Flash, destroying any potentially confidential data.

The current software has a few limitations, for example, only one file is supported and concurrent use of WINC OTA and Host File Download is not supported.

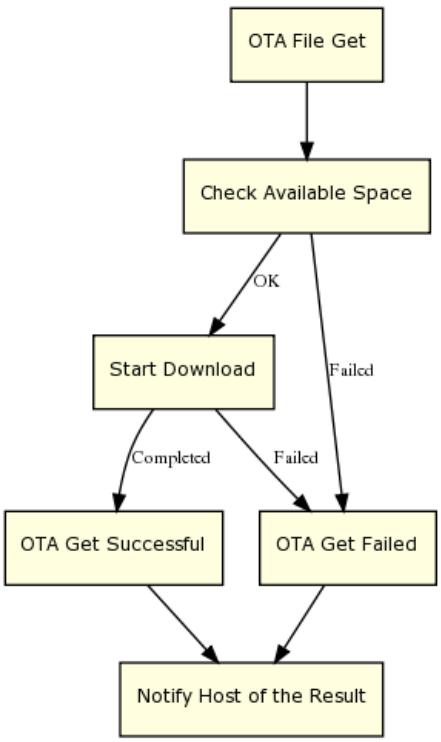

Figure 2 - OTA/ File Download Flow Diagram

Note that the WINC OTA performs additional steps to verify the integrity of the image which are not included in Figure 2 - OTA/ File Download Flow Diagram, this diagram only shows a simplified view of the common steps performed by WINC OTA and Host File Download.

5.7.2 Usage

For a detailed description of the APIs, please refer to *WINC1500_IoT_SW_APIs.chm.*

5.7.3 Downloading a file

The Application can instruct the WINC to download a file from a remote location by calling the API `m2m_ota_host_file_get()`. This call takes the URL which points to the file to be downloaded and a pointer to a callback to be executed once the download finishes. The callback will receive the status of the get operation and on success it will also receive the generated file handler and the total size of the download file.

5.7.4 Reading a File stored in Flash

There are two APIs available to read a file which was stored in the WINC's Flash as a result of a file download. To read via the HIF, the Application can use `m2m_ota_host_file_read_hif()`, which will be non-blocking, causing the host to be interrupted only when data is available, it requires the Application to provide a pointer to a callback to be executed once the requested data is ready. When using `m2m_ota_host_file_read_spi()`, no callback is necessary, since the read will be blocking, but a pointer for a buffer where to store the data will need to be provided.

Reading via SPI will be faster, however it requires the WINC to be put into a special *"download mode"* and then reset once the access to the Flash is no longer required.

The APIs for reading the file will need 3 other parameters which are common to both methods, a valid file handler (valid means matching the internal handler in the WINC), an offset which defines the place where to read from (counting from the start of the file) and the size of the read.

5.7.5 Erasing a File

The Application can instruct the WINC to explicitly erase the file in the WINC's Flash by calling `m2m_ota_host_file_erase()`, this will destroy any data in the Host File partition, which can be useful if the Application is handling confidential information and wants to ensure that it won't be available after use. Erasing the file requires a valid file handler to be passed, but providing a callback to be executed when the erase finishes is optional.

5.7.6 Aborting a File download

If the File download needs to be aborted, the Application can call `m2m_ota_abort()` to stop the download safely. This API is shared with the WINC OTA, so if a WINC OTA is in progress, calling this API will stop it.

5.9 Encrypted Credential Storage

5.9.1 Overview

When a WINC15x0 station successfully connects to a network, connection credentials and session details may optionally be stored in WINC15x0 flash.

This gives rise to a security concern: if a device is captured, an attacker may be able to read the WINC15x0 flash and obtain sensitive information such as the network's PSK, or a user's identity and password.

A new feature in 19.6.1 firmware is that network credentials are encrypted when stored in flash.

Encrypted credential storage is the default behaviour in 19.6.1, but may be overridden by an API option added in the 19.6.1 driver.

5.9.2 Disclaimer

The encryption provided by this feature should not be considered secure; the WINC15x0 does not include a secure flash region.

The encryption is only intended to prevent credentials being revealed in plaintext by an opportunistic read of WINC15x0 flash.

Hence other security practices should still be followed where possible, such as changing passwords regularly and deleting credentials when they are no longer required.

5.9.3 Functionality

When requesting connection to a network, the application can specify how the connection credentials should be stored in WINC15x0 flash. The options are:

- Do not store credentials
- Store credentials unencrypted
- Store credentials encrypted

The credentials consist of:

- SSID
- BSSID (if provided)
- WEP key (for WEP connection)
- Passphrase and PSK (for WPA/WPA2 PSK connection)
- Domain, User name and Password (for WPA/WPA2 1x MSCHAPv2 connection)
- Domain, User name, Certificate and Private Key (for WPA/WPA2 1x TLS connection)

The credentials are stored in WINC15x0 flash when connection succeeds, and only one set of credentials is stored at a time; if new credentials need to be stored then the old credentials are removed (overwritten with 0's).

If credentials are stored in WINC15x0 flash, then the application can request subsequent connections without providing the credentials again, using `m2m_wifi_default_connect`.

If roaming is enabled, roaming can take place regardless of whether the credentials are stored in WINC15x0 flash. (They are stored in data memory for the duration of a connection.)

The application can delete credentials from WINC15x0 flash using `m2m_wifi_delete_sc`.

5.9.4 APIs

The relevant APIs are:

- `m2m_wifi_connect_*` to request connection and store credentials
- `m2m_wifi_default_connect` to request connection using last stored credentials
- `m2m_wifi_delete_sc` to delete stored credentials

Please refer to *WINC1500_IoT_SW_APIs.chm* for full details of these APIs.

5.9.5 Security considerations

5.9.5.1 Encryption details

In order to give an accurate indication of the level of security to be expected, it is necessary to give some detail of the encryption implementation:

- The cipher is AES128.
- The encryption parameters include static elements, device-specific elements, and elements which are randomly generated for each encryption.
- The randomly generated elements are themselves stored in WINC15x0 flash.

5.9.5.2 System design

System designers and application writers should be aware that the WINC15x0 flash is not the only attack point in the system. For example:

- The interface between the WINC15x0 and the host MCU is not secure.
- The host MCU may have its own flash, which may or may not be secure.
- The host MCU may be vulnerable to its code and/or data being read by a debugging tool.

Here are some general recommendations:

- Do not store credentials long-term in the host MCU; instead, rely on `m2m_wifi_default_connect`.
- If the credentials are no longer required, delete them from the WINC15x0 flash using `m2m_wifi_delete_sc`.

5.9.6 Compatibility notes

19.6.1 firmware implements a new format for the WINC15x0 flash store for connection parameters. The effects of this are:

- During a firmware upgrade to 19.6.1, previously stored credentials will be reformatted. After the first successful connection to an access point, these stored credentials will become encrypted.
- During a firmware upgrade to 19.6.1, previously stored IP address and Wi-Fi channel will be deleted.
- After a firmware downgrade from 19.6.1 to previous firmware, credentials stored by 19.6.1 firmware will not be readable by the previous firmware. The operation of the previous firmware is otherwise unaffected.

5.10 Flash Verification Capability (added to flash tools)

5.10.1 Overview

The firmware download process is split into a number of steps. Some of these are optional, for a demonstration of all the options see `download_all.bat` in the 19.6.1 release package.

5.10.2 Existing flash tools

The existing tools operate as follows:

- **image_downloader**
  - Clears all flash (unless **-fw_only** option used)
  - Programs firmware (unless **-skip_fw_update** option used)
  - Configures PLL (unless **-skip_pll_update** option used)
- **gain_builder**
  - Programs one or more gain tables (unless **-skip_programming** option used)
- **tls_cert_flash_tool**
  - Appends new certificates to those in flash (unless **-erase** is used in which case it clears the existing certificates first).
- **root_certificate_downloader**
  - Appends new certificate to those in flash (unless **-e** used in which case it clears the existing certificates first)

5.10.3 New functionality

In 19.6.1, a final verification stage is now performed (in *download_all.bat*):

- **image_cloner**
  - reads flash if **-out_path** used
  - writes flash if **-in_path** used
  - write and read can be combined with an automatic readback verification
  - read/write length can be limited with the **-span** option

The flash verification functionality is based on a virtual flash file, which is modified or created by each of the four tools above when the **-vflash_path** option is used.

With this option set, the tools will perform the same operations on the real flash and virtual flash file.

At the end of the flash process, **image_cloner** reads back the whole flash file and a simple binary file comparison is performed on this file and the virtual flash file to ensure there are no discrepancies.

It is assumed a WINC1510 with 1Mb flash is used, so **image_cloner** is passed a -span of 524288. The top half of flash in the WINC1510 is reserved for downloads so will unlikely match the virtual flash file.

5.11 Built in Automated Test Equipment (ATE) mechanism

A factory flashed WINC15x0 module running the 19.6.1 firmware load will have a special ATE firmware residing in the flash space reserved for OTA transfers (which will be overwritten by the first OTA update).

A host API exists that can be called during WINC initialisation that causes the device to boot into this special firmware (`m2m_ate_init`). The API to control the ATE functions provided by this firmware is detailed in *\ASF\common\components\wifi\winc1500\driver\include\m2m_ate_mode.h*

As an example, the startup calls could be:

```
system_init();
configure_console();
printf(HEADER);
nm_bsp_init();
if(M2M_SUCCESS == m2m_ate_init())
    start_tx_test(M2M_ATE_TX_RATE_1_Mbps_INDEX); // etc
```

There is an example project called <*samd21_xplained_pro_ota_ate_runner*> available from support, which loads the ATE firmware and runs through a number of tests.

If you connect a terminal to the WINC DBG UART, you should see a trace similar to this:

```
- Wifi ATE Firmware Demo --
- SAMD21_XPLAINED_PRO --
- Compiled: May 16 2018 15:06:52 --
(APP)(INFO)Chip ID 1503a0
(APP)(INFO)>>Running TX Test case on CH<11>.
(APP)(INFO)Completed TX Test successfully.
(APP)(INFO)>>Running RX Test case on CH<06>.
(APP)(INFO)Num Rx PKTs: 11134, Num ERR PKTs: 578, PER:
(APP)(INFO)Compeleted RX Test successfully.
(APP)(INFO)Test cases have been finished.
```

5.12 Improved Provisioning Experience

5.12.1 Overview

Improvements have been made to the default provisioning pages and capabilities to ensure an improved user experience with relevant feedback to the user.

5.12.2 HTTP Files

All files found in `<release package>\src\firmware\wifi_v111\src\nmi_m2m\source\http\Server\config\` will be loaded into the HTTP flash section by the **image_builder** tool. The provisioning HTTP server will be able to serve these files when requested by a HTTP client. A few of the files are however referenced by the WINC firmware and should not be moved. There are also a few GET and POST request parameters which perform special actions, like scanning for APs and connecting to an AP. See *Table 4 - Files used during provisioning* for further details.

Table 4 - Files used during provisioning

File	Description
default.html	Main web page
error.json	Served if an error occurs during an AJAX request
ok.json	Served when provisioning credentials have been successfully received
scanresults.json	Serverd with the JSON formatted AP scan results injected at offset 13.

5.12.3 Requesting an AP Scan

Making a GET request to **/?refresh** will initiate an AP scan.

The results of an AP scan are injected into the *scanresults.json* file at offset 13 as it is being served back to the client.

```
{
	"results":[
		{
			"name":"AccessPoint2",
			"mac":"AA:BB:CC:DD:EE:FF",
			"rssi":-40
		},
		{
			"name":"SurveillanceVan",
			"mac":"80:08:13:50:C0:CC",
			"rssi":-61
		},
		{
			"name":"SuperSecretCIA",
			"mac":"CE:EE:EE:EE:E5:55",
			"rssi":-79
		},
		{
			"name":"InconspicuousVan",
			"mac":"CA:69:CB:12:34:56",
			"rssi":-31
		},
		{

		}
	]
}
```

Figure 3 - Example JSON scan response

Each object in the 'results' array contains the AP SSID ('name'), BSSID ('mac') and signal RSSI ('rssi'). The final object in the array is intentionally empty.

5.12.4 Sending Provisioning Credentials

To send provisioning credentials to the WINC, a POST request must be made to the provisioning URL root, containing the following URL encoded parameters:

Table 5 - Provisioning POST parameters

Parameter	Value
connect	Constant:"connect"
login	String: Access Point SSID
password	String: Password
device-name	String: Device name (optional)

Example POST request body:

```
connect=connect&login=MyAccessPoint&password=password123&device-
name=wirelessIOTDevice
```

The response will be a JSON formatted ‘ok’ status if the credentials were successfully received or an ‘error’

```
{"status":"ok"}
```

status:

Or

```
{"status":"error"}
```

Upon receiving provisioning credentials, the WINC notifies the host, which then normally attempts to connect to the provided AP. However, this often causes the WINC to exit provisioning mode too soon, losing the ‘ok’ response. It is recommended to implement a timeout of 1-5 seconds when POSTing the connect request and/or add a delay of 1-5 seconds on the host between receiving credentials and connecting to the AP. If the POST request times out then it should be assumed that the credentials were successfully received and processed.

An error will only occur if the AP name is too short (0 characters) or too long (>32 characters).

A limitation of provisioning mode is that there is no way for the user to know if the WINC managed to successfully connect to the AP via the provisioning web page, since the WINC must first exit provisioning mode before it can attempt to connect to the AP, thus the client losing connection to the provisioning server.

The WINC will remain in idle mode if the provisioned connection fails. The host must be able to handle a failed connection situation, usually by putting the WINC back into provisioning mode.

5.13 Modification of AP mode IP settings

5.13.1 Overview

The DHCP default gateway, DNS server and subnet mask can now be customised when entering AP and provisioning modes. Previously, the default gateway and DNS server would be the same as the host IP of the WINC and the subnet mask would be 255.255.255.0. Configuration of these values allows the use of 0.0.0.0 for the default gateway and DNS server, allowing mobile devices to connect to the WINC AP without disconnecting from the mobile network. Using IPs other than 0.0.0.0 is possible but is generally of no use since only 1 device can connect to the WINC AP at any time.

5.13.2 Usage

To maintain backwards compatibility with older drivers, new structures and APIs have had to be introduced.

To customise these fields when entering AP or provisioning mode the new `tstrM2MAPModeConfig` structure should be populated and passed to the new `m2m_wifi_enable_ap_ext()` or `m2m_wifi_start_provision_mode_ext()` APIs. The `tstrM2MAPModeConfig` structure contains the original `tstrM2MAPConfig` structure for storing the AP SSID, password etc. and another new `tstrM2MAPConfigExt` structure for configuring the default router, DNS server and subnet mask.

For further details and information on the new API calls, refer to *WINC1500_IoT_SW_APIs.chm.*

6 Fixes and Enhancements

This section lists notable fixes and enhancements in 19.6.1 compared to 19.5.4. In addition to these there are many improvements to the overall stability and reliability of the firmware.

6.1 Issues fixed

6.1.1 Firmware fixes

These issues are fixed by upgrading the firmware to 19.6.1:

ID	Description
W1500-59	Incorrect address is provided in M2M_WIFI_REQ_DHCP_CONF callback. The DHCP server address is provided instead of the default gateway address. Fixed: The correct default gateway address is provided.
W1500-27	The firmware crashes if an OTA request is made with a URL which is too long. Fixed: The firmware rejects the OTA request without crashing.
W1500-44	Some TLS socket closure methods result in the socket being leaked in firmware. Fixed: The firmware does not leak TLS sockets.
W1500-101	WPS channel numbering in firmware is incompatible with a 19.4.4 driver. Fixed: The WPS channel numbering in firmware is now compatible with all released drivers, 19.4.4 and later.
W1500-103	The firmware becomes unresponsive if an application requests powersave modes PS_AUTOMATIC or PS_H_AUTOMATIC via a driver prior to 19.5.0. Fixed: The firmware switches off powersave and remains responsive.
W1500-147	DNS name is parsed by firmware as case-sensitive. Fixed: The firmware parses DNS names as case-insensitive.
W1500-152	OTA stalls if the domain fails to resolve. Fixed: The firmware reports an OTA failure.
W1500-179	Configurable tx data rate feature does not work: 1. The rates do not match the `m2m_wifi_conf_auto_rate` API setting. 2. The rate setting is reset during Wi-Fi connection. Fixed: The rate table used by firmware now matches the API setting and the rate setting is not reset during Wi-Fi connection.
W1500-241	The firmware determines authentication policy incorrectly if an AKM suites is present in the authentication response message. Fixed: The firmware determines the correct authentication policy.

W1500-259	In AP mode, the WINC15x0 can get out of sync with the station that is attempting to connect. This happens if the association response gets lost on air. Fixed: The WINC15x0 disconnects the station if its association response is not ACK'd.
W1500-266	Temperature increase causes increased retransmissions. Fixed: PLL is relocked on significant temperature change to maintain TX reliability.
W1500-314	In STA mode, the WINC15x0 sometimes selects WPA when an AP is in mixed (WPA/WPA2) mode. Fixed: The firmware correctly selects WPA2 if an RSN IE is present in the probe response or beacon.

6.1.2 Driver fixes

These issues are fixed by upgrading both driver and firmware to 19.6.1:

ID	Description
W1500-190	The HIF interrupt counter is modified in both task and interrupt context without protection. Fixed: The modification in task context is now done atomically.

6.2 Enhancements

This section covers enhancements to existing features. For new features, see section 5.

6.2.1 Firmware enhancements

These enhancements are available by upgrading the firmware to 19.6.1:

ID	Description
W1500-11	The maximum length of filename for the firmware HTTP server is increased from 16 to 32 characters.
W1500-76	An NTP time update is done once per day, as well as just after connecting to a network.
W1500-119	Default connect functionality is added for WEP and WPA/WPA2 Enterprise connections, as well as WPA/WPA2 PSK and Open connections.
W1500-152	The firmware HTTP client can now access domains that begin with a number.
W1500-287	In the event of DHCP failure in STA mode, the firmware no longer disconnects. Instead, it notifies the host application (using M2M_WIFI_REQ_DHCP_FAILURE) and retries DHCP. Note that the host application only receives the notification if the driver is upgraded to 19.6.1. Legacy drivers may print error messages in the serial trace, such as “REQ Not defined 61” and “host app didn't set RX Done”; these messages are benign and can be safely ignored.

6.2.2 Driver enhancements

These enhancements are available by upgrading both driver and firmware to 19.6.1:

ID	Description
W1500-18	The maximum length of user name and password for WPA/WPA2 Enterprise MSCHAPv2 connections is increased: (i) User name increased from 20 to 132 characters. (ii) Password increased from 40 to 256 characters.
W1500-91	The maximum length of TLS local certificate and key material that the host application can pass to the WINC15x0 is increased from 1588 to 8184 bytes.

7 Terms and Definitions

Term	Definition
AES	Advanced Encryption Standard
AJAX	Asynchronous JavaScript and XML
AKM	Authentication and Key Management
ARP	Address Resolution Protocol
ATE	Automated Test Equipment
BSS	Basic Service Set
CBC	Cyclic Block Chaining
DHCP	Dynamic Host Control Protocol
DHE	Diffie-Hellman Ephemeral
DNS	Domain Name Server
DTIM	Directed Traffic Indication Map
EAP	Extensible Authentication Protocol
EAPOL	EAP Over LAN
ECC	Elliptic Curve Cryptography
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable Read Only Memory
ESD	Electrostatic Discharge
ESS	Extended Service Set (infrastructure network)
HIF	Host Interface
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electronic and Electrical Engineers
JSON	JavaScript Object Notation
MIB	Management Information Base
OTA	Over The Air update
PEAP	Protected Extensible Authentication Protocol
PLL	Phase Locked Loop
PMK	Pair-wise Master Key
PSK	Pre-shared Key
RSA	Rivest-Shamir-Adleman (public key cryptosystem)
RSN	Robust Security Network
RSSI	Receive Strength Signal Indicator
SHA	Secure Hash Algorithm
SNTP	Simple Network Time Protocol
SPI	Serial Peripheral Interface
SSID	Service Set Identifier
TCP	Transmission Control Protocol
TIM	Traffic Indication Map
TLS	Transport Layer Security
TTLS	Tunneled TLS

Term	Definition
WEP	Wired Equivalent Privacy
WINC	Wireless Network Controller
WLAN	Wireless Local Area Network
WMM™	Wi-Fi Multimedia
WMM-PS™	Wi-Fi Multimedia Power Save
WPA™	Wi-Fi Protected Access
WPA2™	Wi-Fi Protected Access 2 (same as IEEE 802.11i)