WINC3400 Software

Release Notes

VERSION : 1.3.1

DATE : 1 JUL, 2019

Abstract

This document presents an overview of the WINC3400 firmware release version 1.3.1, and corresponding driver 1.1.0.

1 Introduction

This document describes the WINC3400 version 1.3.1 release package. This is a release containing Wi-Fi functionality with basic BLE support including an on-chip provisioning profile and custom BLE profiles using the Atmel BLE API and BluSDK.

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

- **Current consumption enhancements**
- **BLE API source code included in release package**
- **WLAN**
  - WPA/WPA2 Enterprise, with the following authentication methods:
    - EAP-TTLSv0/MSCHAPv2
    - EAP-PEAP/MSCHAPv2 (PEAPv0 and PEAPv1)
    - EAP-TLS
    - EAP-PEAP/TLS (PEAPv0 and PEAPv1)
  - New Connection APIs
  - Simple Roaming
- **Network Stack**
  - Dynamic bypass mode
  - Customisable NTP servers
  - TCP keepalive timeout Socket Option
- **TLS Stack**
  - ALPN
  - AES-GCM cipher suites
  - ECDHE-RSA cipher suites
- **Miscellaneous**
  - New APIs for accessing WINC3400 flash from host
  - Encrypted Credential Storage
  - New Flash Tools
  - Built in ATE Mode Support

1.1.2 Other notable changes

- Enhanced user experience with HTTP Provisioning
- AP mode Enhancements
- Flash Double-Write

- Discontinued previous APIs for accessing WINC3400 flash from host (they are replaced by new simpler APIs)

1.2 Firmware readiness

Microchip Technology Inc. considers version 1.3.1 firmware to be suitable for production release.

2 Release summary

2.1 Auditing information

Master Development Ticket : `Sprint:4228`

Wi-Fi:

Release Repository Branch : /chn-vm-svnrepo01.microchip.com/repo/wsg/Wifi_M2M/branches/rel_3400_1.3.1
Subversion Revision : **18052**

BLE:

Release Repository Branch : `/svn/Bluetooth/branches/ATWILC3400_BT_BLE_API`
Subversion Revision : **r7529**

BLE API:

Release Repository Branch : `/svn/Bluetooth/branches/ATWILC3400_BLE_API`
D21 Subversion Revision : **r7477**
SAM4 Subversion Revision : **r7477**

2.2 Version information

WINC Firmware version : 1.3.1
Host Driver version : 1.1.0
Host Interface Level : 1.4

Note: The version reported in the firmware log will be:

```
(0)NMI M2M SW  VERSION 1.3.1
(0)NMI HIF LEVEL (2) 1.4
(0)Built at Jun 28 2019 13:46:26
(0)SVN: 75:18036
```

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:

WINC3400_IoT_SW_APIs.chm

WINC3400_BLE_APIs.chm

2.3.2 Binaries and programming scripts

The main 3400 firmware binary is in the **firmware** directory and named **m2m_image_3400.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_3400.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 1.2.2	Changes in 1.3.1
Wi-Fi STA	

• IEEE 802.11 b/g/n.
• OPEN, WEP security.
• WPA Personal Security (WPA1/WPA2), including protection against key re-installation attacks (KRACK).
	• WPA/WPA2 Enterprise support:
	  ○ EAP-TTLSv0/MSCHAPv2
	  ○ EAP-PEAPv0/MSCHAPv2
	  ○ EAP-PEAPv1/MSCHAPv2
	  ○ EAP-PEAPv0/TLS
	  ○ EAP-PEAPv1/TLS
	  ○ EAP-TLS
	• WPA/WPA2 Enterprise options for phase 1 TLS handshake:
	  ○ Bypass server authentication
	  ○ Specify root certificate
	  ○ Time verification mode
	  ○ Session caching
	• Option to encrypt connection credentials that are stored in WINC3400 flash.
	• Improved connection API, allowing connection via BSSID as well as SSID.
	• Simple Roaming support.

Wi-Fi Hotspot	
• Only ONE associated station is supported. After a connection is established with a station, further connections are rejected. • OPEN and WEP security modes. • The device cannot work as a station in this mode (STA/AP Concurrency is not supported).	• Ability to specify the default gateway, DNS server and subnet mask.
Wi-Fi Direct	
• Wi-Fi direct client is not supported.	No change
WPS	
The WINC3400 supports the WPS protocol v2.0 for PBC (Push button configuration) and PIN methods.	No change
TCP/IP Stack	
The WINC3400 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).	• New socket type “Raw Socket” added, raising the total socket count to 12. • Ability to configure the TCP keepalive settings via Socket Options. • Ability to specify the NTP servers.

Features in 1.2.2	Changes in 1.3.1
• 4 UDP sockets (client or server).	
Transport Layer Security	
The WINC 3400 supports TLS v1.2, 1.1 and 1.0. • Client mode only. • Mutual authentication. • Supported cipher suites are: TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_DHE_RSA_WITH_AES_128_CBC_SHA TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (requires host-side ECC support eg ATECCx08)	• Added ALPN support. • Added cipher suites: TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (requires host-side ECC support eg ATECCx08) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (requires host-side ECC support eg ATECCx08)
Networking Protocols	
DHCPv4 (client/server) DNS Resolver IGMPv1, v2. SNTP	• SNTP servers are fully customisable.
Power saving Modes	
The WINC3400 supports these powersave modes: • M2M_NO_PS • M2M_PS_DEEP_AUTOMATIC	If M2M_PS_DEEP_AUTOMATIC mode is selected the power consumption will be significantly lower than in previous releases, when both BLE and WIFI subsystems are idle
Device Over-The-Air (OTA) upgrade	
The WINC3400 has built-in OTA upgrade. • Firmware is backwards compatible with driver 1.0.8 and later. • Driver is backwards compatible with firmware 1.2.0 and later (though unable to use functionality added since 1.2.0).	No change
Wi-Fi credentials provisioning via built-in HTTP server	
The WINC3400 has built-in HTTP provisioning using AP mode (Open or WEP secured).	• Improved provisioning user experience. • Default gateway and subnet mask can now be customised when in AP mode.
WLAN MAC only mode (TCP/IP Bypass, or Ethernet Mode)	

Features in 1.2.2	Changes in 1.3.1
The WINC3400 does not support WLAN MAC only mode.	• The WINC3400 can be restarted in WLAN MAC only mode, letting the host send/receive Ethernet frames.
ATE Test Mode	
	• Embedded ATE test mode for production line testing driven from the host MCU.
Miscellaneous features	
	• New APIs to allow host applications to read, write and erase sections of WINC3400 flash when the WINC3400 firmware is not running. • Removed previous m2m_flash APIs which allowed access to WINC3400 flash for specific purposes.

3 Test Information

This section summarizes the tests conducted for this release

3.1 Internal testing

Please refer to ticket Jira:W3400-470 for full details.

Testing was performed against the release candidate 1.3.1 against the following configuration(s):

H/W Version	:	WINC3400 XPRO module
Host MCU	:	ATSAMD21-XPRO

For Elliptic Curve cryptography support verification, a CRYPTOAUTH XPLAINED PRO board (containing an ECC508A chip) was inserted into the EXT2 socket on the ATSAMD21-XPRO board.

Testing was performed in both open air and shielded environments.

The following testing has been performed:

1. **General functionality including:**
    1. HTTP Provisioning
    2. BLE API verification
    3. Scanning
        - Active scan
        - Passive scan
    4. Station Mode
        - WPA2/WEP/Open security
        - Throughput testing
    5. AP Mode
        - WPA2/WEP/Open security
        - Throughput testing
    6. IP (TCP and UDP client and server)
    7. WPS
        - PIN
        - Pushbutton
    8. Roaming
        - Channel change and link loss
        - Traversing subnets
    9. 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

10. 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

4. TLS functionality including:

1. RSA cipher suites:
    i. TLS_RSA_WITH_AES_128_CBC_SHA
    ii. TLS_RSA_WITH_AES_128_CBC_SHA256
    iii. TLS_DHE_RSA_WITH_AES_128_CBC_SHA
    iv. TLS_DHE_RSA_WITH_AES_128_CBC_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:
    i. TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256

    Testing uses a NIST standard ECC P256 prime curve server certificate with two chains, one leading back to an ECC root certificate and the other leading to an RSA root certificate.
3. SNI Client Hello extension
4. Server certificate name validation
5. Client authentication

5. Performance under interference

6. TCP/IP stack robustness testing

a. Using an internal implementation of IPerf to verify UDP and TCP throughput scenarios
b. Verification of multi socket functionality
c. HTTP POST/GET

7. Wi-Fi AP interoperability testing

8. Manual Wi-Fi browser interoperability

9. Regression and longevity tests

4 Known Issues

Jira	Severity	Description
W3400-102 (Trac 9299)	Medium	During HTTP provisioning, if applications are running on the device being used to provision the WINC3400, they will not be able to access the internet during provisioning. Furthermore, if they attempt to do so, then the WINC3400 can become flooded with DNS requests and crash. This applies to HTTP provisioning only; BLE provisioning is unaffected. Also, this only applies if powersave is enabled. Recommended workarounds: (1) Use M2M_NO_PS when WINC3400 is in HTTP provisioning mode. (2) Close other internet applications (browsers, skype etc) before HTTP provisioning. If crash occurs, reset system via system_reset(). Alternatively, m2m_wifi_reinit() can be used to reset just the WINC. In this case, the different driver modules also need to be initialized (m2m_ota_init(), m2m_ssl_init(), socketInit()).
W3400-40	Medium	The WINC3400 occasionally fails to proceed with 4-way handshake in STA mode, when using 11N WPA2. It does not send M2 after receiving M1. Recommended workaround: Retry the Wi-Fi connection.
W3400-293	Medium	1% of Enterprise conversations fail due to the WINC3400 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.
W3400-298	Medium	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.
W3400-461 (Trac 8085)	Low	Sometimes the WINC3400 fails to see ARP responses sent from certain APs at 11Mbps. Recommended workaround: None. The ARP exchange will be retried several times and the response will eventually get through to the WINC3400.
W3400-60 (Trac 8212)	Low	During BLE provisioning, the AP list is not cleaned up at the start of each scan request. As a result, the AP scan list can sometimes display duplicate or old scan entries. Recommended workaround: Only use one scan request during BLE provisioning.
W3400-59 (Trac 8436)	Low	APIs at_ble_tx_power_get() and at_ble_max_PA_gain_get() return default values which do not correspond to the actual gain settings. Recommended workaround: None. Do not use these APIs.

W3400-30 (Trac 8858)	Low	If the TLS server certificate chain contains RSA certificates with keys longer than 2048 bits, the WINC takes several seconds to process it. A Wi-Fi group rekey occurring during this time can cause the TLS handshake to fail. Recommended workaround: Retry opening the secure connection.
W3400-64 (Trac 9290)	Low	at_ble_tx_power_set() needs special handling. Return values 0 and 1 should both be interpreted as successful operation. Refer to WINC3400_BLE_APIs.chm for more detail. Recommended workaround: Process the return value with care, according to the API documentation.
W3400-240	Low	After writing new firmware to the WINC3400, the first Wi-Fi connect attempt in STA mode takes an extra 5 seconds. Recommended workaround: Allow longer for the Wi-Fi connection to complete.
W3400-451	Low	When running in AP mode, the WINC3400 DHCP Server sometimes takes 5 to 10seconds to assign an IP address. Recommended workaround: Allow longer for DHCP to complete.

5 New Features

There are several new features added since the previous released version (1.2.2).

5.1 PowerSave enhancements

5.1.1 Automatic BLE PowerSave

Prior to 1.3.1 the BLE subsystem was placed into low power mode using the `m2m_wifi_req_restrict_ble` and `m2m_wifi_req_unrestrict_ble` APIs. These API calls are now obsolete as the BLE subsystem will automatically enter low power mode whenever possible. Calling these APIs with 1.3.1 firmware on the WINC will have no effect.

BLE relies on an external RTC to enter Idle powersave, the presence of this will be auto detected and if not present, BLE will be prevented from getting into powersave mode, even when idle.

5.1.2 System level PowerSave

In addition to the above, prior to 1.3.1 the WINC3400 Wi-Fi subsystem would only enter powersave mode when connected to an 802.11 network and the powersave mode was set to `M2M_PS_DEEP_AUTOMATIC`.

In 1.3.1 the concept of system level powersave has been introduced whereby when both the Wi-Fi and BLE subsystems are idle, the WINC3400 will always automatically enter the lowest power state.

Therefore, when `M2M_PS_DEEP_AUTOMATIC` powersave mode is operational and the WINC is IDLE (both BLE and Wi-Fi are inactive), current consumption will drop considerably (from around 77mA to <1mA).

5.2 WPA(2) Enterprise Connectivity

Support for WINC3400 Wi-Fi connection in STA mode with WPA/WPA2 Enterprise has been introduced in 1.3.1.

5.2.1 Authentication Methods

Here is a list of authentication methods supported by WINC3400:

- 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.

5.2.1.1 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 WINC3400 client, trusted root certificates are stored in flash. These can be written by a tool during production, or by the customer application at runtime. It is recommended that the application specify a particular root certificate via the Enterprise TLS options API (see section 5.2.4).

5.2.1.2 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 WINC3400 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).

5.2.2 Connection Caching

Caching significantly reduces reconnection time, which is especially important during roaming (see section 5.5).

Here is a list of the caching methods supported by WINC3400.

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

5.2.2.1 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 WINC3400 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.

5.2.2.2 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 WINC3400 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.

5.2.3 Credential Handling

5.2.3.1 Domain

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

WINC3400 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 username and domain, including separators, is limited to a maximum of 100 characters.

5.2.3.2 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.

WINC3400 connection APIs provide an option to specify whether "anonymous" or the actual identity is used during phase 1.

5.2.3.3 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.

WINC3400 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.

5.2.4 Enterprise TLS Options

All Enterprise authentication methods use a TLS handshake in phase 1 (the single phase in the case of EAP-TLS).

WINC3400 set/get option APIs allow the application to configure some details of this TLS handshake.

5.2.4.1 Bypass Server Authentication

This option allows server authentication to be bypassed during Enterprise connection. Bypassing server authentication is not recommended as it allows the phase 2 data to be sent to a rogue server.

5.2.4.2 Specify Root Certificate for Server Authentication

This option allows the server to be verified against a particular root certificate out of the many which may be present in the WINC3400 flash store. This is recommended in order to truly verify the server.

5.2.4.3 Time Verification Mode

This option allows configuration of the mode for checking expiry of the server certificate chain. The WINC3400 would normally obtain the current time via Wi-Fi connection, so certificate expiry checking during the first connection is problematic.

5.2.4.4 TLS Session Caching

This option allows TLS session resume capability to be enabled or disabled. See section 5.2.2.2.

5.2.5 Storing Credentials in WINC3400 Flash

The credentials and TLS options can optionally be stored in WINC3400 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 5.9.

5.2.6 APIs

Please refer to *WINC3400_IoT_SW_APIs.chm* for full details of these APIs and their parameter types.

5.2.6.1 Options

Here are the APIs that are available to the application for setting/getting Enterprise TLS options:

- `m2m_wifi_1x_set_option` for setting TLS options for future Enterprise connection attempts.
- `m2m_wifi_1x_get_option` for getting the current state of Enterprise TLS options.

Here are the available Enterprise TLS options:

- `WIFI_1X_BYPASS_SERVER_AUTH` for bypassing server authentication.
- `WIFI_1X_TIME_VERIF_MODE` for modifying time verification of server certificates.
- `WIFI_1X_SESSION_CACHING` for enabling/disabling TLS session resume.
- `WIFI_1X_SPECIFIC_ROOTCERT` for specifying a particular root certificate for server authentication.

5.2.6.2 Connection

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 EAPPEAPv1/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 with the same TLS options (assuming a previous connection request used the option to store the credentials in WINC3400 flash).

5.2.7 Implementation Detail

The WINC3400 implementation follows the authentication method specifications given in Table 1 - Authentication method specifications. For EAP-PEAPv0 and EAP-PEAPv1, there are some intentional deviations from the specifications in order to improve interoperability with existing server implementations.

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

Table 1 - Authentication method specifications

5.2.7.1 EAP-PEAPv1 Deviations

The WINC3400 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).

5.2.7.2 EAP-PEAPv0 Deviations

The WINC3400 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”).

5.2.8 Test Summary and Interoperability

The WINC3400 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 the following servers:

- Cisco server (Aironet 2600 series)
- Microsoft server
- Aruba server
- Ubiquiti server

There is information online about some differences between different server implementations. Based on this, it is expected that the WINC3400 client implementation would be interoperable with a wide range of servers, with the exceptions of “Radiator” PEAPv1 and “Lucent NavisRadius” PEAPv1.

5.3 TLS Application Layer Protocol Negotiation

Support for Application Layer Protocol Negotiation (ALPN) has been added for SSL sockets. The main utility of this is to allow customer applications to use HTTP/2 over TLS.

The feature is available via new socket APIs `set_alpn_list` and `get_alpn_index`.

5.3.1 Using the ALPN feature

To use ALPN with an SSL socket, the customer application must call `set_alpn_list` before connecting, then `get_alpn_index` after the socket connection succeeds.

5.3.1.1 `set_alpn_list`

The parameters to `set_alpn_list` are the socket id and a list of one or more application layer protocols, in preference order.

The application layer protocols are represented by IANA strings, defined at https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml#alpn-protocol-ids

The list provided to `set_alpn_list` must consist of the appropriate IANA strings, separated by spaces.

For example, for HTTP/2 over TLS (1st preference) or HTTP/1.1 (2nd preference), the list provided to `set_alpn_list` should be: “h2 http/1.1” (including NUL terminator). i.e:

```
0x68 0x32 0x20 0x68 0x74 0x74 0x70 0x2f 0x31 0x2e 0x31 0x00
```

5.3.1.2 `get_alpn_index`

The parameter to `get_alpn_index` is the socket id. The return value indicates which application layer protocol has been negotiated:

- 1: The negotiated protocol is the first one in the list that was provided to `set_alpn_list`. (In the above example, this would be HTTP/2 over TLS.)
- 2: The negotiated protocol is the second one in the list that was provided to `set_alpn_list`. (In the above example, this would be HTTP/1.1.)
- etc for return values greater than 0.
- 0: No negotiation occurred, for example because the TLS peer did not support ALPN.

Note that if negotiation occurs unsuccessfully (i.e. the peer does not support any of the protocols listed by the customer application), then the socket connection fails. The customer application can determine the cause of failed socket connections using the socket API `get_error_detail`.

5.3.2 ALPN APIs

Further details of the APIs can be found at *WINC3400_IoT_SW_APIs.chm*.

5.4 New Connect APIs

This section refers to changes in driver version 1.1.0. If 1.3.1 firmware is used with a driver older than 1.1.0, then this section can be ignored.

New APIs have been added in the 1.1.0 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 driver 1.1.0:

- `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 *WINC3400_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 WINC3400 flash (the default behavior of 1.3.1 firmware is to encrypt credentials. See section 5.9 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 usernames 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 Simple Roaming

5.5.1 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 1.3.1, the WINC3400 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 WINC3400 roams from one AP to another AP, both of which are inside the same IP subnet. Layer 3 roaming occurs when the WINC3400 roams from one AP to another AP which are in different subnets, whereby the WINC3400 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 it 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 WINC3400 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`.

5.5.2 Usage

5.5.2.1 Enabling roaming

The API call `m2m_wifi_enable_roaming` sets the WINC3400 to detect link-loss more aggressively, and once detected, to perform the roaming steps specified in 5.5.1. For full details, see WINC3400_IoT_SW_APIs.chm.

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

5.5.2.2 Disabling roaming

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

5.6 Dynamic bypass mode

5.6.1 Overview

Bypass mode can now be enabled via Host MCU using an API instead of requiring an entirely different firmware build, as it was the case in releases prior to 1.3.1. Bypass mode enables the WINC to send/receive Ethernet frames and pass them to/from the IP stack running on the Host MCU.

5.6.2 Usage

5.6.2.1 Enable bypass mode

In order to use bypass mode, there are a few parameters of `tstrEthInitParam` that need to be configured and then passed onto `m2m_wifi_init`/`m2m_wifi_reinit`. First, the parameter `u8EthernetEnable` should be set to `M2M_WIFI_MODE_ETHERNET` and a callback to handle the events on the Ethernet interface should be created and set via parameter `pfAppEthCb`.

The receive buffer and its size will also need to be configured via the API `m2m_wifi_set_receive_buffer`, prior to using the WINC in bypass mode.

5.6.2.2 Callback

The callback for bypass mode is of type `tpfAppEthCb` and expects 3 parameters, `u8MsgType`, the message type, `pvMsg`, a pointer to the data and `pvCtrlBuf`, a control structure to record how much data has been sent to the host and how much data has been delivered at any given time. Received frames will be delivered via this API.

5.6.2.3 Set the packet receive buffer

To be able to receive frames in bypass mode and get them successfully delivered to the host, the receive buffer needs to be configured by calling `m2m_wifi_set_receive_buffer`, this accepts two arguments, a pointer to the buffer to hold the data, which is allocated and controlled by the host, and the maximum size of this buffer.

5.6.2.4 Send a packet

In order to send a frame in bypass mode, `m2m_wifi_send_ethernet_pkt` API should be used, which will expect a pointer to the buffer holding the frame and the frame size as parameters.

5.7 Customisable NTP Servers

5.7.1 Overview

In releases prior to 1.3.1, 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 firmware version 1.3.1, the NTP server(s) used by the WINCs built in SNTP client can now be customised using the `m2m_wifi_configure_sntp` API. The new configuration options allow not only for the use of a single NTP server, but also 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 provided an NTP 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 shortest period between SNTP client re-sync attempts is one day and it is triggered by a DHCP renew event. For example, if the DHCP renewal period is 1 hour, then the 48th DHCP renewal will trigger an NTP update (note that the DHCP client renews every half the specified period). However, if the DHCP renewal period is 4 days, then the time will be re-synced every 2 days.

Each NTP server is attempted 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 *WINC3400_IoT_SW_APIs.chm* for details of how to use the `m2m_wifi_configure_sntp` *API.*

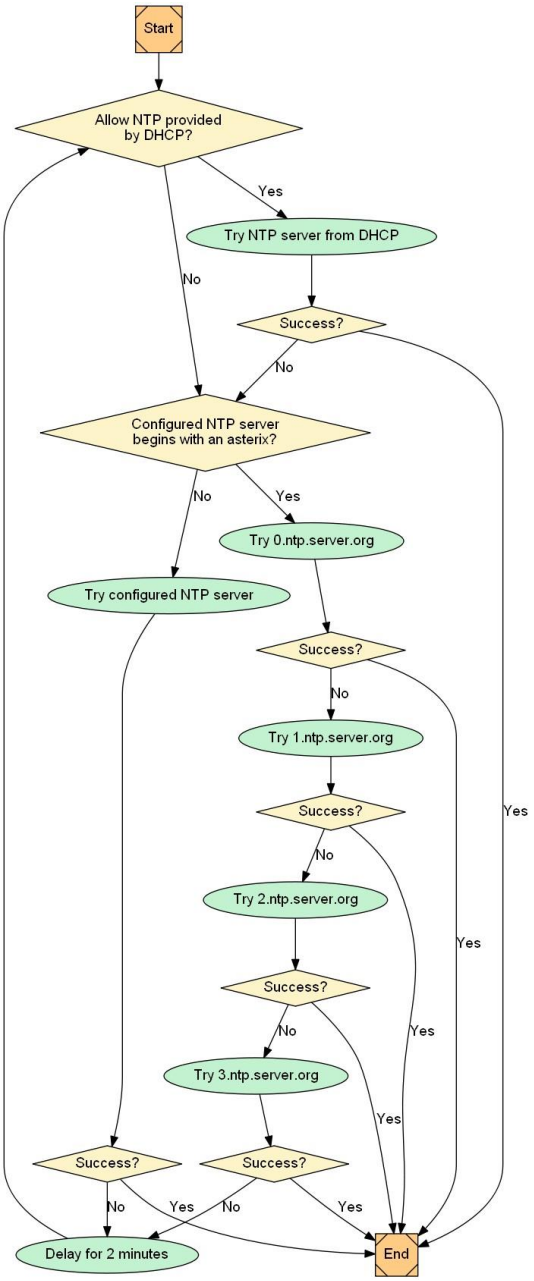

Figure 1 - WINC SNTP states

5.8 TCP keepalive timeout Socket Option

5.8.1 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.

The setting will only take effect if it is set after calling `connect` or `bind` for the socket.

Table 2 - TCP Keepalive socket options

Option	Description	Default Setting
SO_TCP_KEEPALIVE	Enable or disable keepalive for a TCP socket	Enabled
SO_TCP_KEEPIDLE	Duration between two keepalive transmissions in idle condition	60 seconds
SO_TCP_KEEPINTV	Duration between two successive keepalive retransmissions, if ack to the previous keepalive transmission is not received	0.5 seconds
SO_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 *WINC3400_IoT_SW_APIs.chm.*

5.6.2 Examples

Table 3 - TCP keepalive examples

Call	Description

`int32 val = 200;`
`setsockopt(sock, 0, SO_TCP_KEEPIDLE, &val, sizeof(val));`
`val = 10;`
`setsockopt(sock, 0, SO_TCP_KEEPINTV, &val, sizeof(val));`
`val = 5;`
`setsockopt(sock, 0, SO_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_TCP_KEEPALIVE, &val, sizeof(val));`	Disable TCP keepalive for a socket

5.9 Encrypted Credential Storage

5.9.1 Overview

When a WINC3400 station successfully connects to a network, connection credentials and session details may optionally be stored in WINC3400 flash.

This gives rise to a security concern: if a device is captured, an attacker may be able to read the WINC3400 flash and obtain sensitive information such as the network's PSK, or a user's identity and password. A new feature in 1.3.1 firmware is that network credentials are encrypted when stored in flash.

Encrypted credential storage is the default behaviour in 1.3.1, but may be overridden by an API option added in the 1.1.0 driver.

5.9.2 Disclaimer

The encryption provided by this feature should not be considered secure; the WINC3400 does not include a secure flash region.

The encryption is only intended to prevent credentials being revealed in plaintext by an opportunistic read of WINC3400 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 WINC3400 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, Username and Password (for WPA/WPA2 1x MSCHAPv2 connection)
- Domain, Username, Certificate and Private Key (for WPA/WPA2 1x TLS connection)
- TLS options (for WPA/WPA2 1x connection)

The credentials are stored in WINC3400 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 WINC3400 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 WINC3400 flash. (They are stored in data memory for the duration of a connection.)

The application can delete credentials from WINC3400 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 *WINC3400_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 WINC3400 flash.

5.9.5.2 System design

System designers and application writers should be aware that the WINC3400 flash is not the only attack point in the system. For example:

- The interface between the WINC3400 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 credentials are no longer required, delete them from the WINC3400 flash using `m2m_wifi_delete_sc`.

5.9.5.3 Compatibility notes

1.3.1 firmware implements a new format for the WINC3400 flash store for connection parameters. The effects of this are:

- During a firmware upgrade to 1.3.1, 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 1.3.1, previously stored IP address and Wi-Fi channel will be deleted.
- After a firmware downgrade from 1.3.1 to previous firmware, credentials stored by 1.3.1 firmware will not be readable by the previous firmware. The operation of the previous firmware is otherwise unaffected.

5.10 Raw Socket

5.10.1 Overview

Typically, in order to be able to send/receive a frame in raw format, the WINC3400 needs to be put in Bypass Mode, see 5.6 for more details. The Raw Socket feature allows for a socket to be created which will be able to send/receive frames in raw format while the WINC is running in normal mode (not in Bypass Mode).

This functionality is particularly interesting to send host assembled frames and to be able to process and reply frames which are IP fragmented, since the WINC does not support IP fragmentation and these frames end up processed in the host.

The current limitation of this feature is that it can only send and receive ICMP frames, if enabled. For UDP/TCP frames, these can only be sent, any UDP/TCP frames received won't ever reach the host via this socket in raw format.

5.10.2 Usage

Firstly, the socket module will need to be initialized via `socketInit` and a socket of type Raw needs to be created in order to be able to send raw frames, this can be achieved by calling the API socket with parameters `AF_INET`, `SOCK_RAW`, `SOCKET_FLAGS_IPPROTO_RAW`.

A call for the `bind` API with a raw socket is not required, however, if issued, it will return success.

In order to receive ICMP packets as raw on the host, the application must set the ICMP filter option for the raw socket via `setsockopt`. The current implementation supports either a filter none (0) or a filter all (any other value), if the filter is set to none, then all ICMP frames will be reported to the host as raw frames and via raw socket, otherwise, ICMP packets will be internally handled by the WINC.

When sending raw frames, the application will need to build the frame and call `sento` while passing the buffer which contains the frame and specifying the size of the frame. The destination address can also be passed onto the `sendto` API and if it passed via the API, it will overwrite the destination address field in the buffer passed.

Frames are received via `recvfrom` and handled via the socket handler callback registered with the socket module by calling `registerSocketCallback`.

5.11 Changed Flash Access APIs

This section refers to changes in driver version 1.1.0. If 1.3.1 firmware is used with a driver older than 1.1.0, then this section can be ignored.

New APIs have been added in the 1.1.0 driver to allow the host application to access sections of WINC3400 flash.

The flash access APIs in previous versions of the driver are no longer available. The discontinued APIs are: `m2m_flash_init`, `m2m_flash_get_state`, `m2m_flash_updateimage`, `m2m_flash_read_image` and `m2m_flash_rootcert_*`.

5.11.1 New APIs

Here is a list of the APIs which have been added in driver 1.1.0:

- `m2m_flash_erase_sector`
- `m2m_flash_write`
- `m2m_flash_read`
- `m2m_flash_switch_firmware`

Please refer to *WINC3400_IoT_SW_APIs.chm* for full details of these APIs and their parameter types.

5.12 New Flash Tools

5.12.1 Overview

The flash tools, which are typically included in the firmware release package, were updated in the 1.3.1 release.

The previous tools were designed to be run sequentially to build up a complete image in the flash, the new tools collect the various elements into an image file which is then loaded to the flash in a single operation.

The image file is created under Windows, but it may be loaded to flash with either Windows or Linux as the Operating System support was extended from Windows only to Windows, Linux and Mac.

The new tools use a configuration file (**flash_image.config**), which can be found under the ***/*firmware** folder, inside the release package provided. In this configuration file are defined several different sections of the WINC flash, where two must be highlighted, *"gain table"* and *"root certificates"*, which are described in sections 5.12.2 and 5.12.3, respectively.

The new tools to build and load the flash image have command line help, however there is a file provided **prepare_image.cmd** which gives a primer in their use.

5.12.2 Flashing the gain table

The gain table should be specified in **flash_image.config** (under section *"gain table"*), however, in order to maintain backwards compatibility, csv files can still be used and can be automatically converted (via **prepare_image.cmd**) to the correct format prior to writing it to the flash.

5.12.3 Flashing certificates

In order to flash the necessary certificates, these are no longer required to be copied to a particular folder, instead, the path to the certificate should be specified in **flash_image.config***, under "root certificates"*. This means any customer can simply maintain a certificate library and customise which certificates are flashed based on their requirements, which provides extra flexibility. However, this also means that it is no longer sufficient to just copy the certificates into the ***/*root_certificate_downloader** in order to flash the certificates, as it is mandatory to list them in **flash_image.config** to be picked up by the flash tool.

5.13 Built in Automated Test Equipment (ATE) mechanism

5.13.1 Overview

A factory flashed WINC3400 module running the 1.3.1 firmware 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\WINC3400\driver\include\m2m_ate_mode.h*. As an example, the startup calls for a TX test could be:

```
system_init();
configure_console();
printf(STRING_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 several tests (TX and RX).

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 14 2019 08:52:42 --
(APP)(INFO)Chip ID 3400d2

(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)Completed RX Test successfully.

(APP)(INFO)Test cases have been finished.
```

6 Fixes and Enhancements

These are the fixes and enhancements since the previous released version (1.2.2).

6.1 Issues fixed in Wi-Fi.

JIRA ID	Description
W3400-435	Only check for events if Wi-Fi has started to avoid host getting stuck.
W3400-365	After ARP timeout (5 seconds), don't disable ARPs, restart the timer.
W3400-205	Ignore unknown AKM suites when parsing Auth Policy in RSN/WPA IE
W3400-162	Limit WEP key size.
W3400-203	Disconnect the MAC when an ASOC_RSP in AP mode is failed to be ACKed by the STA, to avoid getting out of sync.
W3400-380	Fix potential memory leak when updating the connection state.
W3400-357	Ensure data piggybacked on the ACK in the TCP 3-way Handshake is processed.
W3400-196	Make sure transmissions are suspended when loading new gain settings
W3400-126	Fixed issue with HTTP client where domains that begin with a number would be treated as an IP, and an issue with OTA getting stuck if the domain fails to resolve.
W3400-80	Fixed issue where after supplying an invalid OTA URL the device will get stuck returning OTA_STATUS_INVAILD_ARG even for valid URLs.
W3400-330	Scan requests result in active scans even when passive is requested
W3400-72 (Trac 9292)	Race condition when BLE is active can cause BLE to get stuck and hold the radio indefinitely.

6.2 Issues fixed in BLE

JIRA ID	Description
W3400-466	Failure to advertise if the advertising data and scan response data are the same
W3400-407	BLE firmware can lock up when switching between advertising and connected modes
W3400-405	(BLE API) Changes to accommodate BLE API source code release
W3400-182	(BLE API) More consistent reporting of events
W3400-183	(BLE API) Correct unpacking of messages using interface macros
W3400-241	Fix for possible lockup during BLE pairing
W3400-171	Fix possible BLE lockup during switch to Wi-Fi sleep.

6.3 Enhancements

6.3.1 Firmware enhancements

JIRA ID	Description
W3400-202	Ensure WPA2 is always chosen as the preferred security protocol if a choice is presented by the AP
W3400-136	Always ensure we have at least 1 free receive buffer when receiving UDP traffic
W3400-378	HttpClientParseURL now strips off the port number from the host name (or IP address).
W3400-116	Increased maximum file name length for HTTP server from 16 to 32.
W3400-83	Reworked Wi-Fi provisioning to use AJAX to request JSON formatted data when refreshing devices list and connecting to an AP instead of inserting tables into the HTML. User Interface/Experience improved.
W3400-278	Enable GCM. Cipher suite ECDHE-RSA-AES128-GCM-SHA256 is now supported.
W3400-73	Add ability to specify custom DNS and default router IP addresses in AP mode
W3400-165	Add ability to specify subnet mask in AP mode
W3400-129	Set both BLE and Wi-Fi LUT to same value, which improves channel 10 EVM
W3400-94	Updated gain table
W3400-430	Implement Flash double erase/write to alleviate marginal erase/program

6.3.2 Driver enhancements

JIRA ID	Description
W3400-274	Add WINC state tracking to help host identify if Wi-Fi has started (eg for Flash access).
W3400-107	Better hif level checks and error handling
W3400-323	Change driver to locally store ping application callback instead of passing it over the HIF. Also, pings must now be 1-out-1-back; if the application requests a second ping then the first ping will not trigger the callback.
W3400-401	M2m_flash enhancements

6.3.3 Other improvements

General performance and stability improvements.

7 Terms and Definitions

Term	Definition
AES	Advanced Encryption Standard
ARP	Address Resolution Protocol
BLE	Bluetooth Low Energy
BSS	Basic Service Set
CBC	Cyclic Block Chaining
DHE	Diffie-Hellman Ephemeral
DNS	Domain Name Server
DTIM	Directed Traffic Indication Map
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)
GAP	Generic Access Profile
HTTP	Hypertext Transfer Protocol
IBSS	Independent BSS (ad-hoc network)
IEEE	Institute of Electronic and Electrical Engineers
MIB	Management Information Base
MQTT	Message Queuing Telemetry Transport
NDIS	Network Driver Interface Specification
OTA	Over The Air update
PCI	Peripheral Component Interconnect
PMK	Pair-wise Master Key
PSK	Pre-shared Key
RSA	Rivest-Shamir-Adleman (public key cryptosystem)
RSN	Robust Security Network
SHA	Secure Hash Algorithm
SPI	Serial Peripheral Interface
SSID	Service Set Identifier
RSSI	Receive Strength Signal Indicator
TIM	Traffic Indication Map
TLS	Transport Layer Security
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)