CRATON SDK

4.11.0 Release Notes

Rev. 1.0

October 2016

Revision History

Rev.	Date	Authors	Reviewers	Content
1.0	25 Oct 2016	B. Moosa	O.Ashery	Initial revision

Table of Contents

Revision History 2
Table of Contents 3
List of Figures 5
List of Tables 6
1. Introduction 7
1.1 Who Should Update? 7
1.2 Abbreviations 7
1.3 References 8
1.4 Release Details 9
1.4.1 Versioning 9
1.4.2 SDK Variants 10
1.4.3 Release Content 12
1.5 Download Information 12
2. Change List 13
2.1 New Features 13
2.2 Bug Fixes 14
2.3 Other changes 14
3. Known Issues 15
3.1 IEEE WAVE 15
3.1.1 AT-5486: Channel Switching 15
3.2 ETSI ITS-G5 15
3.2.1 AT-6874: V2X Sample API 15
3.2.2 AT-5766: CAN Transmission 15
3.2.3 AT-5692: V2X Transmission 15
3.2.4 AT-6047: Logging to Console 16
3.3 PANGAEA Device 16
3.3.1 AT-1936: Indication LEDs 16
4. Upgrade Guide 17
4.1 Overview 17
4.2 Flashing Guide 17
4.2.1 Using NOR Flash API 17
4.2.2 Using CLI command 18
4.2.3 Using U-Boot 20
4.2.4 MIB Config Files 20
4.3 SDK Upgrade 22
4.3.1 Developers of CRATON Firmware 22
4.3.2 SECTON mode Developers 23

5. Appendix 24
5.1 Changes in Examples 24
5.1.1 remote-posix/gnss 24

List of Figures

Figure 1 – SDK tarball example 12
Figure 2 – CLI command ‘firmware update’ 18
Figure 3 – Flashing via 'firmware update' 19
Figure 4 – Right-click on wlanConfigSaveStatus and choose “Set…” 21
Figure 5 – Set value of wlanConfigSaveStatus to "save (2)" 22

List of Tables

Table 1: Abbreviations 7
Table 2: Firmware variants 10
Table 3: PANGAEA boards 11
Table 4: Release content 12

1. Introduction

Autotalks provides a SDK for developing firmware for its V2X chipset, consisting of CRATON V2X Communication Processor and PLUTON V2X RF Transceiver.

The SDK also supports Infineon SLx97 HSM running Autotalks firmware and STMicroelectronics STA80 family of GNSS Receivers running firmware configured by Autotalks.

This document details the history of changes and upgrade procedures.

1.1 Who Should Update?

Updating is recommended to all users of Autotalks V2X platform.

As a general rule, issues found in preceding minor releases (see section 1.4.1) will only be fixed in the latest SDK as a patch release or in subsequent minor SDK release.

1.2 Abbreviations

Abbreviations used in this document.

Table 1: Abbreviations

Abbreviation	Definition	Description
AES	Advanced Encryption Standard	
CAN	Controller Area Network	
CBC-MAC	Cipher Block Chaining Message Authentication Code	
CBR	Channel Busy Ratio	The percentage of time during which the channel was busy in the last probing interval
DCC	Decentralized Channel Congestion	
DR	Dead Reckoning	
DRAW	Dead Reckoning Automotive Way	Refers to STMicroelectronics DR for Teseo-II/III devices
DSRC	Dedicated Short Range Communication	
ECC	Elliptic Curve Cryptography	
ECIES	Elliptic Curve Integrated Encryption Scheme	
ECDSA	Elliptic Curve Digital Signature Algorithm	
EPD	EtherType Protocol Discrimination	
GNSS	Global Navigation Satellite System	
HSM	Hardware Security Module	A device used for safe storage of security keys
HTTP	HyperText Transfer Protocol	
IMQ	Inter-processor Message Queue	Used for inter-processor communication (mc SDK)

LLC	Link Layer Control	
MC	Multi Core	Refers to multi core CRATON SDK variant
MIB	Management Information Base	
NMEA	National Marine Electronics Association	GNSS receiver messages as defined in NMEA 0183 Std
NVM	Non Volatile Memory	
OID	Object Identifier	Uniquely identifies a managed object in MIB hierarchy
PKI	Public Key Infrastructure	
PoTi	Position and Time	
PPS	Pulse Per Second	Electrical pulse emitted on round seconds
RBF	ROM Bootable Flash	A complete flash layout which complies to boot ROM rules
RBI	ROM Bootable Image	An image which can be booted from ROM
RBIS	Secure ROM Bootable Image	RBI which is signed by OEM key
RF	Radio Frequency	
RSVC	Remote Services	Refers to the services available on host CPU when using “sectonm” SDK (or running in SECTON mode)
RX	Reception	Refers to receptions of packets over any medium
SC	Single Core	Refers to single core CRATON SDK variant
SCMS	Security Credential Management System	PKI used in US
SDK	Software Development Kit	
SHA	Secure Hash Algorithm	
SOU	Sensor Over Uart	Proprietary STMicroelectronics NMEA messages used to inject sensor data into Teseo DRAW.
SNMP	Simple Network Management Protocol	
TAI	Temps Atomique International	Seconds since 2004-01-01T00:00:00Z, including leap seconds
TSF	Timing Synchronization Function	Refers to CRATON’s PPS-aligned counter
TX	Transmission	Refers to transmission of packets over any medium
UTC	Universal Time, Coordinated	Seconds since 1970-01-01T00:00:00Z, excluding leap seconds
V2V	Vehicle-to-Vehicle	
V2X	Vehicle-to-X	X may be vehicle, infrastructure or portable device
WLAN	Wireless Local Area Network	

1.3 References

References cited in this document.

[1] IEEE, 1609.2-2016 Security Services for Applications and Management Messages, 2016-05.

[2] SAE, J2945/1 System Requirements for V2V Safety Communications D5.0, 2015-XX.

[3] C2C-CC, Basic System Profile v1.1.0, 2015-12.

[4] IEEE, 1609.4-2016 Multi-Channel Operation, 2016-05.

[5] IEEE, 1609.3-2016 Networking Services, 2016-05.

1.4 Release Details

1.4.1 Versioning

Autotalks SDK version strings consist of major, minor and patch numbers, and ends in a release stage. Version number and release stage are explained below (see **Error! Reference source not found.**).

1.4.1.1 Version number

- **Major:** Refers to CRATON SDK generation; CRATON SDK is currently in its 4th and last generation (SDK 5.x will support Autotalks CRATON2 and SECTON devices).
- **Minor:** A running number starting at 0; incremented for the "next" SDK release, typically scheduled every ~3 months.
- **Patch:** A running number starting at 0; incremented for releases which resolve critical bugs or add urgent features (which cannot wait until the next minor release).

1.4.1.2 Release stage

- **Alpha:** Releases which are incomplete and only partially tested; labeled "alphaX", where X is a running number starting at 1.
- **Beta:** Releases which are feature-complete and under SW, QA and RF testing; labeled "betaY", where Y is a running number starting at 1.
- **Release candidate:** Releases which are feature-complete and well tested; labeled "rcZ", where Z is a running number starting at 1.
- **Final release:** A final stable release; labeled "rel" (without a number).

1.4.1.3 Definitions

Release-related definitions used in this document.

- A "minor" release is a final release, with patch number equal to 0.

- A “patch” release is a final release, with a patch number larger than 0.

1.4.1.4 How to retrieve SDK version?

CRATON SDK version can be retrieved via:

1. Getting MIB attribute sysDescr[1] (MIB file mibs/SNMPv2-MIB.mib) over SNMP.
2. Function mib_get_sysDescr (header file atlk/mibs/snmpv2-mib.h).
3. Issuing command version in SDK CLI prompt.
4. SDK version is printed to console during system initialization.

1.4.2 SDK Variants

The following table details the variants of firmware included in this release.

Table 2: Firmware variants

	Variant Name	Description
1	Single Core SDK (termed **sc** SDK)	Firmware built using this SDK uses CRATON’s main core only – an ARM Cortex R4F CPU (ARC CPUs are disabled). User has two options: 1. User code runs on ARM CPU. 2. User code runs on external CPU (termed host CPU) or in a virtual machine; CRATON emulates SECTON V2X add-on (see additional details in section 1.4.2.1).
2	Multi Core SDK (termed **mc** SDK)	Firmware built using this SDK uses all three of CRATON’s cores – ARM Cortex R4F and two ARC 625D CPUs. User code runs on all three cores. Inter-core communication is done over Inter-processor Message Queue (IMQ).
3	SECTONm[2] SDK (termed **sectonm** SDK)	This SDK contains a pre-built (i.e. “closed”) firmware image to be used on CRATON (users cannot build firmware). User code runs on external CPU or in a virtual machine; CRATON emulates SECTON V2X hardware add-on (see additional details in section 1.4.2.1).

1.4.2.1 SECTON mode

The term “SECTON mode” refers to CRATON running in a mode in which it emulates SECTON V2X add-on, and in which the user’s code is running on host CPU (whether real or virtual).

[1] OID: 1.3.6.1.2.1.1.1 (textual OID: iso.org.dod.internet.mgmt.mib-2.system.sysDescr)
[2] SECTONm firmware can only run on a board with SRAM, such as board type atk22017c-sectonm.

The differences between using sc and sectonm SDK in SECTON mode:

- CRATON uses DRAM when using sc SDK, it uses SRAM when using sectonm SDK.
- The image "ref.img" supplied with sc SDK initializes additional components which are not relevant to SECTON (e.g. VCA daemon, navigation service), but these components are mostly harmless (i.e. they do not change the behavior of code running above libsectonhost library).
- The sectonm SDK only includes headers, examples and files relevant to SECTON mode; sc SDK also includes headers, examples and files relevant for target development.

1.4.2.2 Board types

CRATON SDK supports a wide range of board types. For each board type, the SDK contains:

1. Object "ref-sys.o" which includes board-specific[3] hardware initialization. This object will be linked in to user image depending on the BOARD variable set in Makefiles (see section "Building a CRATON firmware example for a specific board" in the SDK user manual).
2. Image "ref.img" – this image must be flashed on CRATON when running in SECTON mode; otherwise it is included as a pre-built image for the convenience of users who do not write code (e.g. for the purpose of conducting field tests or laboratory measurements).

Board names starting with "atk" are boards designed at Autotalks while boards starting with "pcb" were designed by Autotalks customers. Board names ending in sectonm are boards which use SRAM (rather than DRAM), emulating next gen SECTON V2X hardware add-on.

Board names of PANGAEA boards (or compatible customer boards) are detailed in Table 3.

Table 3: PANGAEA boards

[3] Using an incompatible image might result in a hang system during boot or in hardware which does not function properly; in rare cases, it might result in physical damage to the hardware.

	Board Type	Description
1	atk22016	PANGAEA4 EVK, all revisions
2	atk22017a	PAGAAEA4-S EVK, revisions A and B
3	atk22017c	PANGAEA5 EVK (with integrated Skyworks RF front end)
4	atk22017c-sectonm	PANGAEA5 EVK, on which SRAM is assembled and sectonm SDK is used

1.4.3 Release Content

The following table details the contents of this release.

Table 4: Release content

	Item	Description
1	SDK tarballs (files ending in tar.xz).	XZ-compressed tar archives (one per each SDK variant). File naming convention is explained in **Error! Reference source not found.**
2	SHA256 hash files (files ending in sha256.txt)	SHA256 hash of tarballs, for the purpose of verifying its integrity (e.g. via Linux sha256sum[4] command)
3	User manual	The starting point for development using CRATON SDK
4	API manual	Auto-generated API manual (doxygen-based, from headers; users are encouraged to use the headers directly)
5	Release Notes	This document

1.4.3.1 SDK tarball

The following figure explains SDK tarball naming conventions.

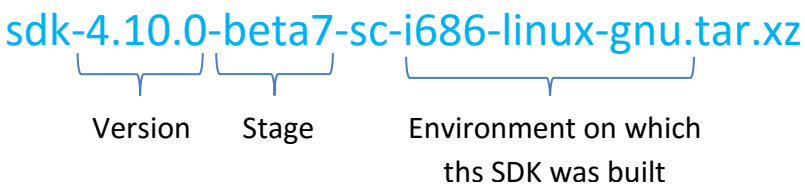

Figure 1 – SDK tarball example

1.5 Download Information

This release can be downloaded via the below link.

- **URL:** *https://files.secureserver.net/0fI5qQzZWel05z*
- **Pass:** J1Rv$de

[4] See ***sha256sum*** man page: *http://linux.die.net/man/1/sha256sum*

2. Change List

The following change list is relative to SDK 4.10.3.

This release contains a couple of system-wide change that developers must take into account:

- `TX_TICK_RATE` was changed from 1000 to 2000. This means: Do not assume that 1 timer tick is equal to 1 ms. This applies to "`tx_thread_sleep`", "`tx_mutex_get`", "`tx_semaphore_get`" and any other ThreadX function which uses "`wait_option`" represented by timer ticks.

2.1 New Features

- Increased ThreadX tick rate.
- V2X API now supports 1609.4 channel switching. The support is an addition to the API, not a change. In other words, applications that don't need the 1609.4 channel switching API, do not need to change/align their current V2X API code. Current V2X API behavior remains unchanged.
- Secure boot support. Please refer to *CRATON_ROM_bootloader.pdf* for more information.
- WAVE IPv6 support, in conjunction with 1609.4 channel switching.
- Support of remote services over raw Ethernet with example
- DHCP client support with example
- Remote HSM support
- BroadR-Reach Ethernet PHY support
- Memory/CPU utilization alarm support with example
- Expose getters and setters for syslog sink and level
- Deprecate irrelevant channel switching MIB attributes
- Enable/Disable WFP estimator via boot parameter 'wfp_estimator'
- Phantom compensator support
- WLAN EPD support
- Add start/stop API for the following modules:

  - SNMP
  - FTP
  - CLI
- Added board type trace, as part of system initialization
- Added CRATON CPU clock (240/360Mhz) trace, as part of system initialization
- WLAN MAC level CRC on RX packets support
- VCA over TI CC3100 Wi-Fi AP support for atk23010 board types
- UART0 baud-rate is configured based on U-boot parameter '`baudrate`'

2.2 Bug Fixes

- Fix RF GPIO configuration for pcb101v1 board
- Fix faulty Ethernet PHY configuration on atk23010 boards
- Default RF frequency should be according to channel band
- DCOC: Remove writing to LNA mixer during periodic calibration

2.3 Other changes

- Deprecate BSM and WSMP code from VCAD and VCA-MIB
- Improve non-descriptive trace related to FS partition mount failure
- CLI ping command: Add suggestion to disable Firewall on host side
- Deprecate VSA support
- Deprecate manually-written sections in API
- New signal mapping configuration for atk22036b
- Increase RAM FS to accommodate 6 MB file for pcb101v1 board

3. Known Issues

The section details notable known issues only. Issue numbers (of the form “AT-NUM”) are from internal Autotalks issue tracking system (in most cases only parent issue is mentioned).

3.1 IEEE WAVE

3.1.1 AT-5486: Channel Switching

- Channel switching table in mibs/AUTOTALKS-WLAN-MIB.mib (wlanCsTable) cannot be used to implement 1609.4-2016.
- RF performance (e.g. functionality of DCOC algorithm) in channel switching is sub-optimal, which may result in PER.

3.2 ETSI ITS-G5

3.2.1 AT-6874: V2X Sample API

- V2X sample API is not yet implemented in ***libsectonhost*** library; specifically, subscribing to receive CBR samples is not supported.
- This means that ETSI DCC is not supported in practice when using SDK in “SECTON mode” or when using sectonm SDK. System Performance

3.2.2 AT-5766: CAN Transmission

- CAN transmission above ~500 frames/sec will result in buffer allocation errors.
- This will remain a known issue as this figure represents acceptable performance given expected use-cases.
- Mitigation is possible via a different allocation of CAN HW TX and RX buffers and usage of CAN HW filters (at RX). Contact Autotalks support for further info.

3.2.3 AT-5692: V2X Transmission

- V2X transmission above ~400 frames/sec will result in buffer allocation errors.
- This will remain a known issue as this figure represents acceptable performance given expected use-cases.

- Mitigation is possible via enlargement of the V2X packet pool (see craton/v2x_config.h), though this will generally only enable larger bursts.

3.2.4 AT-6047: Logging to Console

- Non-watchdog panic can be triggered during mass logging to console.
- This will remain a known issue – users are expected to use UDP sink for massive logging (see section “Using the system logger” in SDK user manual).

3.3 PANGAEA Device

3.3.1 AT-1936: Indication LEDs

- PANGAEA device LEDs are not configured.

4. Upgrade Guide

4.1 Overview

This guide is divided into two sections:

1. **Flashing guide** – its main audience are users which use PANGAEA EVK (or Autotalks partner EVKs) for field and interoperability testing, and users developing on host CPU in SECTON mode (using sc or sectonm SDK variants).
2. **SDK upgrade guide** – for developers writing code which runs on target (sc and mc SDK variants) or on host CPU (sc and sectonm SDK variants).

4.2 Flashing Guide

Two methods of firmware flashing are supported on (non-SECURE[5]) CRATON devices:

1. **NOR flash API** – enables flashing of firmware image located in RAM to user specified partition[6] on NOR flash. A system reboot is required for the new firmware to load.
2. **U-Boot** – natively supports commands for flashing NOR flash devices. Flashing via U-Boot[7] requires knowledge of the address and size of the main firmware image partition (on the specific board type being flashed).

Autotalks recommends using **NOR flash API**. For convenience, a CLI command is included in the SDK (from SDK 4.9.x) for performing the flashing procedure (the command is implemented above NOR Flash API).

Section 4.2.4 details the behavior of the system related to **MIB config files**, when firmware update is done from firmware which was built using different SDK versions, and what to do about it.

4.2.1 Using NOR Flash API

Example of using the API is available in the firmware update example in Autotalks CRATON SDK (examples/craton-threadx/firmware/fw-update-example.c).

The main steps of flashing via the NOR Flash API:

[5] Flashing of SECURE CRATON devices is outside the scope of this document.
[6] Main firmware image partition is supported in SDK 4.10.0; other partitions are not supported.
[7] This guide assumes Autotalks CRATON boot has already been flashed on CRATON device; how to flash boot for the first time is outside the scope of this document.

1. Download firmware image to RAM via any means implemented by the user (e.g. over Ethernet, 802.11p, CAN bus, from microSD card, or any other accessible I/O).
2. Read the partition table via norfl_part_table_read and validate that there is enough space for the image in the partition.
3. Verify that the image is a valid firmware image via fw_uimage_valid[8].
4. Flash the image on chosen partition via norfl_part_rewrite.

4.2.2 Using CLI command

Autotalks SDK includes a CLI command implemented above NOR flash API for the purpose of simplifying its usage for end-users who do not write code.

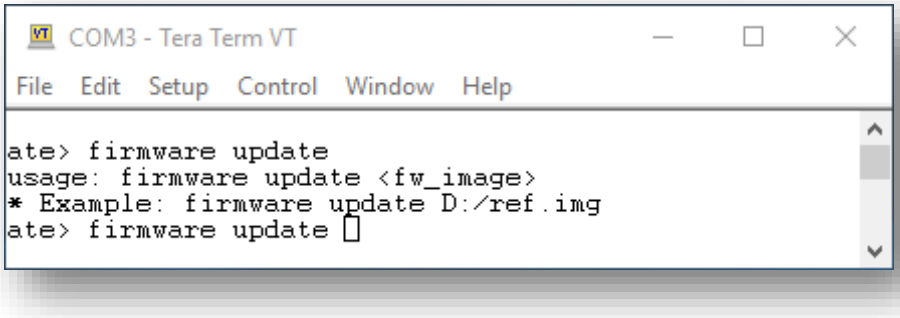

Figure 2 – CLI command 'firmware update'

4.2.2.1 Command arguments

Figure 2 details CLI command "firmware update" arguments, where:

- ***fw_image***: Path to firmware image; can be on any FS partition supported on the device:
- **A:/** – microSD[9]
- **B:/** – NOR flash
- **C:/** – NAND flash[9]
- **D:/** – RAM
- **E**:/ – USB mass storage[9]

[8] Not relevant for image used on SECURE CRATON devices.
[9] Supported on select board types.

4.2.2.2 Usage Example

1. Load the firmware binary to RAM partition over FTP[10] via a FTP client application of your choice (such as FileZilla[11]).
2. To flash the firmware (assuming its name is "ref.img"), issue CLI command:

   ```
   firmware update D:/ref.img
   ```

3. Upon first reboot, U-Boot[12] automatically updated the "filesize" environment variable (please see expected traces in Figure 3).

Figure 3 shows the expected output[13] of flashing followed by a system reboot.

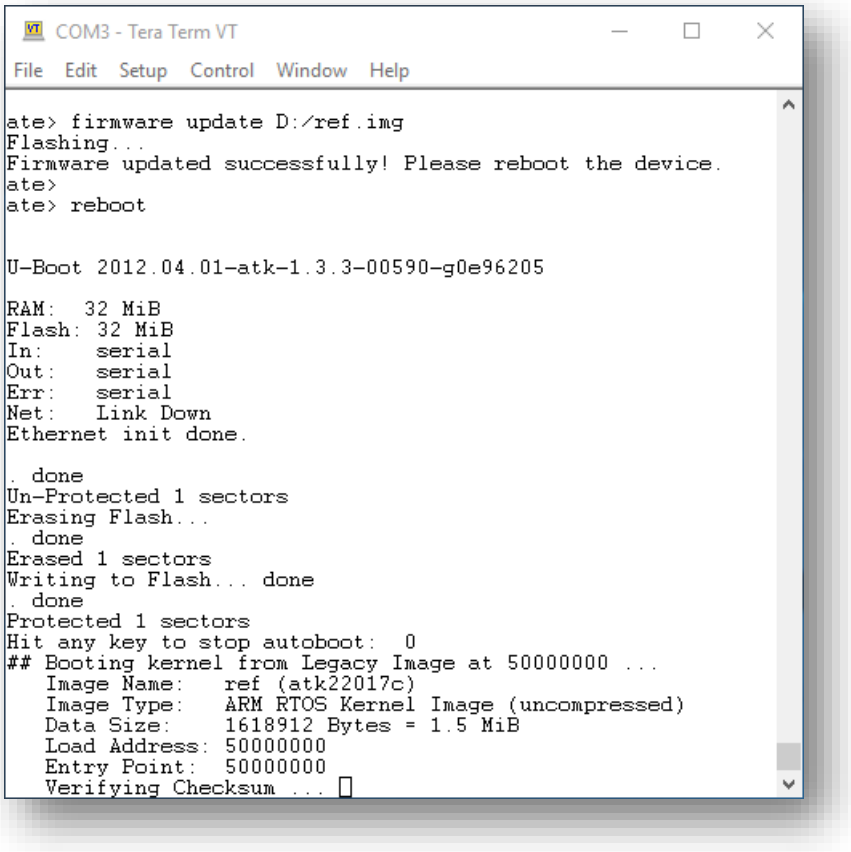

Figure 3 – Flashing via 'firmware update'

[10] FTP server included with CRATON SDK supports "binary mode" only.
[11] FileZilla homepage: *https://filezilla-project.org/*
[12] Only supported from boot 1.2.10 and when environment variable "bootcmd" is set with an expansion macro for environment variable "filesize" (i.e. bootcmd has the form: "*cp.b 80000 50000000 ${filesize}; bootm*"). In other cases, "bootcmd" must be manually updated by user.
[13] Output might vary depending on SDK version, configured Syslog sink and log level.

4.2.3 Using U-Boot

Flashing using U-Boot is covered in section “Flashing CRATON firmware image” in the SDK user manual.

Note: Loading the image into RAM can also be done over UART via Kermit protocol; how to do so is detailed in “CRATON-Boot-1.3.3-Release-Notes.pdf” (details will be added to this document in future version – TBD).

4.2.4 MIB Config Files

MIB configuration files are stored on NOR flash at "B:\NAME-mib.conf", where NAME is one:

- “wlan” – MIB attributes of mibs/AUTOTALKS-WLAN-MIB.mib.
- “nav” – MIB attributes of mibs/AUTOTALKS-NAV-MIB.mib.
- “vca” – MIB attributes of mibs/AUTOTALKS-VCA-MIB.mib.

Note that values of MIB attributes with “MAX-ACCESS” set to “read-write” only are stored.

A MIB configuration file contains the MIB's revision at time of saving; auto-generated[14] code is always aligned with current MIB revision at time of release. Note that MIB configuration files are forward and backward compatible by design.

4.2.4.1 **MIB config traces**

When firmware was built with a SDK in which MIB revision is different than the revision stored in MIB config file, the following is expected:

- Revision mismatch between MIB configuration file and SDK revision - a warning of the following form is output: “Header attribute mismatch. Expected 201510130000, received 201508100000”.
- When a MIB attribute is not defined, not writable, deprecated, or obsolete - a warning of the following form is output: "Attribute ‘wlanPresetFrequency0’ not defined, not writable, deprecated, or obsolete”, Where:
- **Not defined** – due to downgrading to a SDK in which the attribute did not exist.
- **Not writable** – due to using a SDK in which the attribute's "MAX-ACCESS" is "read-only".

[14] Storage and loading of MIB attribute values is done via auto-generated code from MIB.

- **Deprecated or obsolete** – due to using a SDK in which the MIB attribute's "status" is "deprecated" or "obsolete".
- When the MIB attribute's value is invalid - an error of the following form is output and default value is used: “mib_set_wlanCsFrequencyA: Unsupported ‘value’ 0”.
- When a MIB attribute with "status" equal to "read-write" does not exist in configuration file - default value is used (no warning is output).

4.2.4.2 MIB config saving

To remove the warnings/errors, please re-save the relevant MIB configuration file.

For example, to save configuration of mibs/AUTOTALKS-WLAN-MIB.mib via SnmpB, right-click on the attribute (see Figure 4) and sets its value to “save (2)” (see Figure 5).

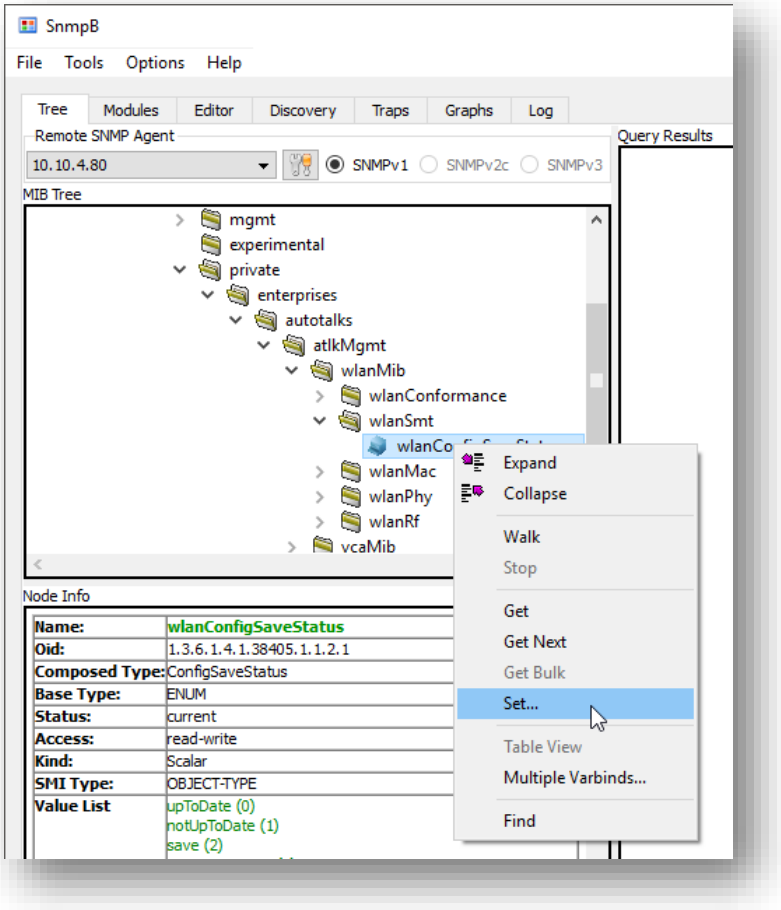

Figure 4 – Right-click on wlanConfigSaveStatus and choose “Set…”

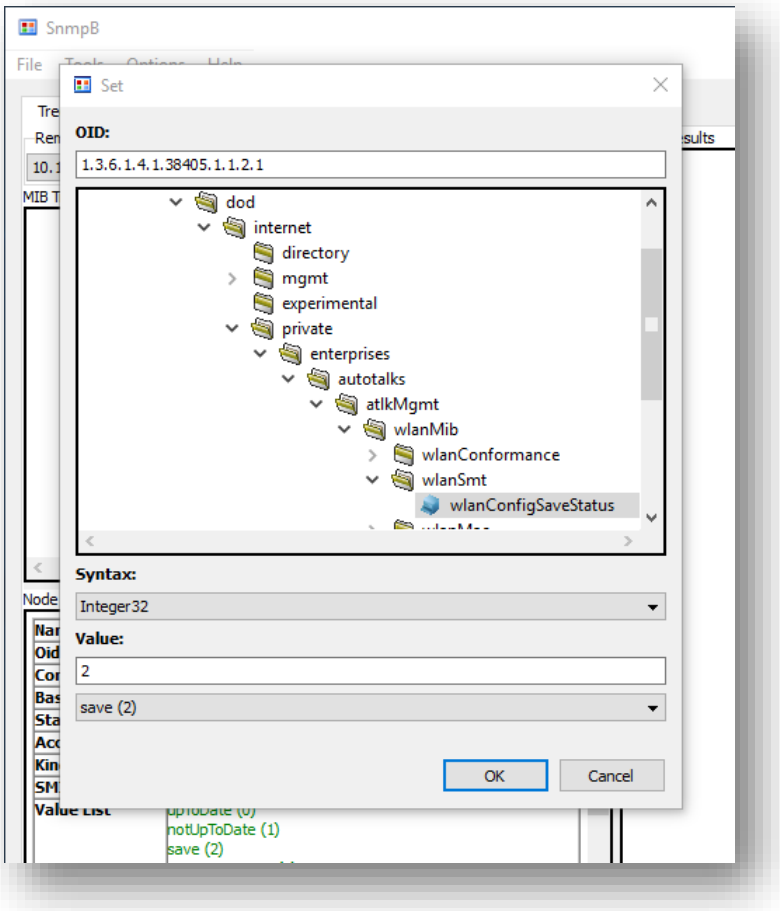

Figure 5 – Set value of wlanConfigSaveStatus to "save (2)"

4.3 SDK Upgrade

Users which have followed the SDK user manual, have one or more installations of preceding SDKs at "/tools/autotalks/SDK_NAME/" (SDKs can be installed side-by-side).

Upgrading is done differently by users which are developing code for target or host:

4.3.1 Developers of CRATON Firmware

1. Install SDK tarball via the procedure detailed in section "Installing SDK" of the user manual.
2. Update the variable "SDK_DIR" in your Makefiles to point to the location of the new SDK.

- Align you code with any changes introduced in SDK APIs (see section **Error! Reference source not found.** for a list of changes introduced in this SDK).

4.3.2 SECTON mode Developers

1. Flash the image "ref.img" matching the board type[15] you are using via the procedures detailed in section 4.2.
2. Install SDK tarball via the procedure detailed in section "Installing SDK" of the user manual, including compilation of libsectonhost library.
3. Update the variable "SDK_DIR" in your Makefiles to point to the location of the new SDK.
4. Align you code with any changes introduced in SDK APIs (see section **Error! Reference source not found.** for a list of changes introduced in this SDK).

4.3.2.1 Ported libsectonhost

If the host library was ported to a different OS in preceding SDK, users must manually merge changes between previous host library, current host library and user changes. How to do so is outside the scope of this document.

[15] For example, when using PANGAEA5, flash the image contained under "board/atk22017c/arm/img/ref.img".

5. Appendix

5.1 Changes in Examples

This section details additions and changes in SDK examples.

5.1.1 remote-posix/gnss

- Add missing ‘break’ in remote GNSS example

Contact Information

Home page:

http://www.auto-talks.com/

Headquarters (Israel)

Grand Netter Building

P.O.B. 3846, Kfar-Netter Israel, 40593

Phone: (+972) 9-886-5300

Fax: (+972) 9-886-5301

info@auto-talks.com