TRUSTED® COMPUTING GROUP

SPECIFICATION

Measurement and Attestation RootS (MARS) Serialization Interface Specification

Version 1
Revision 0
January 29, 2024

Contact: admin@trustedcomputinggroup.org

PUBLISHED

DISCLAIMERS, NOTICES, AND LICENSE TERMS

ACKNOWLEDGEMENTS

The TCG would like to gratefully acknowledge the contributions of the following individuals and their organizations, who volunteered their time and efforts for the development of this specification.

Tom Brostrom, Cyber Pack Ventures, Inc.

Eoin Carroll, Toyota Motor North America

Mariza Marrero, United States Government

Vadim Sukhomlinov, Google LLC

Dick Wilkins, Phoenix Technologies, Inc.

CONTENTS

TABLES

FIGURES

1 SCOPE

This document is the Measurement and Attestation RootS (MARS) Serialization Interface Specification. It defines the data encoding of commands and parameters received by and responses sent from a MARS instance over a transport interface (e.g., SPI, UART, I2C, network) that requires data to be serialized. Proper serialization of data ensures that a deserializing receiver can reassemble a copy that is semantically identical to that of the sender. This specification is transport protocol agnostic. The MARS Serialized Architecture diagram is included from [1], and shown in Figure 1.

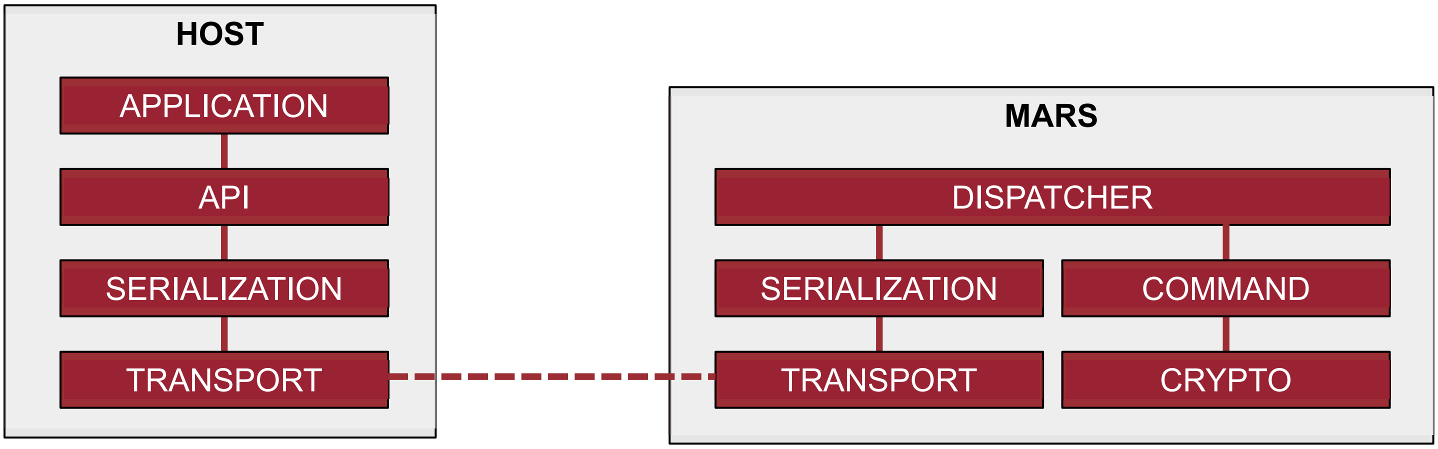

Figure 1 - MARS Serialized Architecture

1.1 Key Words

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” in this document form normative statements and are to be interpreted as described in RFC-2119, *Key words for use in RFCs to Indicate Requirement Levels*.

1.2 Statement Type

Please note a very important distinction between different sections of text throughout this document. There are two distinctive kinds of text: informative comment and normative statements. Because most of the text in this specification will be of the kind normative statements, the authors have informally defined it as the default and, as such, have specifically called out text of the kind informative comment. They have done this by flagging the beginning and end of each informative comment and highlighting its text in gray. This means that unless text is specifically marked as of the kind informative comment, it can be considered a kind of normative statement.

EXAMPLE: Start of informative comment

This is the first paragraph of 1–n paragraphs containing text of the kind *informative comment* ...

This is the second paragraph of text of the kind *informative comment* ...

This is the nth paragraph of text of the kind *informative comment* ...

To understand the TCG specification the user must read the specification. (This use of MUST does not require any action).

End of informative comment

2 Abbreviations, Acronyms and Terms Used

AK	Attestation Key
API	Application Programming Interface
CBOR	Concise Binary Object Representation
CDDL	Concise Data Definition Language
I2C	Inter-Integrated Circuit serial communication bus
MARS	Measurement and Attestation RootS
PCR	Platform Configuration Register
PS	Primary Seed
SHA	Secure Hash Algorithm
SPI	Serial Peripheral Interface
TCG	Trusted Computing Group
TSR	Trusted Sensor Register
UART	Universal Asynchronous Receiver-Transmitter

3 Formatting

Serialized I/O with MARS SHALL be formatted according to RFC 8949 – Concise Binary Object Representation (CBOR). CBOR "is a data format whose design goals include the possibility of extremely small code size, fairly small message size, and extensibility without the need for version negotiation" [2]. Implementations SHALL comply with CBOR Core Deterministic Encoding Requirements, using the shortest tag possible. Note that this specification does not use tagging to extend CBOR data types.

Concise Data Definition Language (CDDL) [3] is used in Appendix A to define the formatting of MARS commands (as *mars_command*) and responses (as *mars_response*). Informative examples of CBOR-encoded MARS commands and responses are in Appendix B. Informative examples of using CDDL to verify MARS command and response formatting are in [4].

3.1 Data Type Mappings Between MARS, CBOR, and CDDL

3.1.1 Aggregation

Individual command parameters and response values are referred to as data items. A sequence of one or more data items is contained within a single CBOR array, denoted in this specification using CDDL square brackets. All serialized commands to and responses from MARS SHALL be formatted using CBOR arrays. Arrays MUST NOT have an insufficient nor excess number of data items.

3.1.2 Integers

All MARS integer data types are unsigned and denoted in this specification as CDDL *uint*.

3.1.3 Register Selection

A MARS regSelect 32-bit bitmask SHALL be encoded as a CDDL uint.

3.1.4 Byte Arrays

All MARS byte arrays SHALL be encoded as byte strings, denoted in this specification as CDDL *bstr*.

3.1.5 Boolean

The Boolean type used by MARS is denoted in this specification as CDDL *bool*.

3.1.6 Null Pointer

A null pointer is denoted in this specification as CDDL *nil*.

3.2 Commands

An array of data items in a command to MARS SHALL begin with a command code encoded as a uint. Values for command codes are defined in [1] by macros prefixed by MARS_CC_. Each command code SHALL be followed by zero or more encoded parameters required for that command.

Before beginning the actions associated with a command, a set of command format and consistency checks SHALL be performed by MARS. These checks are listed here, and SHOULD be performed in the indicated order.

1) do not start execution of command before all parameters are parsed,
2) verify that the data is a CBOR array,
3) unmarshal a command code and verify that the command is implemented,
4) verify that the number of data items in the array matches the number of parameters required by the command,
5) unmarshal parameters and verify that their types are correct
   a) verify that the bstr lengths are supported by the implementation and match constraints defined by the MARS' profile, and return MARS_RC_VALUE otherwise,
   b) verify that ranges of values are supported, and return an appropriate response code (e.g., if PCR selection indicates an unimplemented register, return MARS_RC_REG).

3.3 Responses

An array of data items in a response from MARS SHALL begin with a response code encoded as a uint. Values for response codes are defined in [5] by macros prefixed by MARS_RC_. The response code SHALL be followed by zero or more encoded output values from and required by the corresponding command.

4 Bibliography

[1] TCG, "MARS API Specification, Version 1, Revision 2," 9 May 2023. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-mars-api-specification/.

[2] IETF, "RFC 8949 Concise Binary Object Representation (CBOR)," [Online]. Available: https://www.rfc-editor.org/rfc/rfc8949.html.

[3] IETF, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures," [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8610.

[4] TCG, "MARS Repository," [Online]. Available: https://github.com/TrustedComputingGroup/MARS.

[5] TCG, "MARS Library Specification, Version 1, Revision 14," 2 Jan 2023. [Online]. Available: https://trustedcomputinggroup.org/resource/mars-library-specification/.

Appendix A CDDL for MARS Commands and Responses

```
; Implementation specific parameters
; These parameters must be set by the implementor according to the
; relevant profile specification.
uint16 = 0..65535
; Arbitrary length binary data
binary_data = bstr .size (0..2048)
; binary data of length PT_LEN_DIGEST
digest_data = bstr .size (16..64)
; binary data of length PT_LEN_KSYM
ksym_data = bstr .size 32
; binary data of length PT_LEN_KPUB
kpub_data = bstr .size 32
; binary data of length PT_LEN_SIGN
sign_data = bstr .size (16..64)
; bit mask selecting PCRs and TSRs
reg_select_type = uint .size 4
; index of a PCR or TSR
reg_index_type = uint .size 1
; End of implementation specific parameters

; MARS Command Codes
CC_SelfTest =            0
CC_CapabilityGet =       1
CC_SequenceHash =        2
CC_SequenceUpdate =      3
CC_SequenceComplete =    4
CC_PcrExtend =           5
CC_RegRead =             6
CC_Derive =              7
CC_DpDerive =            8
CC_PublicRead =          9
CC_Quote =               10
CC_Sign =                11
CC_SignatureVerify =     12

; MARS Response Codes
rc_success = 0
rc_io      = 1
rc_failure = 2
rc_buffer  = 4
rc_command = 5
rc_value   = 6
rc_reg     = 7
rc_seq     = 8
```



```
; MARS Capability Property Tags
; number of consecutive PCRs implemented on this MARS
PT_PCR =        1
; number of consecutive TSRs implemented on this MARS
PT_TSR =        2
; size of a digest that can be processed or produced
PT_LEN_DIGEST = 3
; size of signature produced by CryptSign()
PT_LEN_SIGN =   4
; size of symmetric key produced by CryptSkdf()
PT_LEN_KSYM =   5
; size of asymmetric key returned by PublicRead()
PT_LEN_KPUB =   6
; size of private asymmetric key produced by CryptAkdf()
PT_LEN_KPRV =   7
; TCG-registered algorithm for hashing by CryptHash()
PT_ALG_HASH =   8
; TCG-registered algorithm for signing by CryptSign()
PT_ALG_SIGN =   9
; TCG-registered algorithm for symmetric key derivation by CryptSkdf()
PT_ALG_SKDF =   10
; TCG-registered algorithm for asymmetric key derivation by CryptAkdf()
PT_ALG_AKDF =   11

SelfTest = (
    CC_SelfTest,
    full_test: bool,
)

SelfTest_Rsp = (
    rc_success /
    rc_failure
)

CapabilityGet = (
     CC_CapabilityGet,
     capability:    PT_PCR /
                    PT_TSR /
                    PT_LEN_DIGEST /
                    PT_LEN_SIGN /
                    PT_LEN_KSYM /
                    PT_LEN_KPUB /
                    PT_LEN_KPRV /
                    PT_ALG_HASH /
                    PT_ALG_SIGN /
                    PT_ALG_SKDF /
                    PT_ALG_AKDF,
)
```



```
CapabilityGet_Rsp = (
    (rc_success,
    capability_data: uint16) /
    rc_value /
    rc_buffer
)

SequenceHash = (
     code: CC_SequenceHash,
)

SequenceHash_Rsp = (
    rc_success /
    rc_seq /
    rc_command /
    rc_failure
)

SequenceUpdate = (
     CC_SequenceUpdate,
     binary_data,
)

SequenceUpdate_Rsp = (
    (rc_success,
    binary_data) /
    rc_seq /
    rc_buffer /
    rc_command /
    rc_failure
)

SequenceComplete = (
     CC_SequenceComplete,
)

SequenceComplete_Rsp = (
    (rc_success,
    binary_data) /
    rc_seq /
    rc_buffer /
    rc_command /
    rc_failure
)

PcrExtend = (
    CC_PcrExtend,
```



```
    reg_index: reg_index_type,
    digest_data,
)

PcrExtend_Rsp = (
    rc_success /
    rc_reg /
    rc_buffer /
    rc_command /
    rc_failure
)

RegRead = (
    CC_RegRead,
    reg_index: reg_index_type,
)

RegRead_Rsp = (
    (rc_success,
    digest_data) /
    rc_reg /
    rc_buffer /
    rc_command /
    rc_failure
)

Derive = (
    CC_Derive,
    reg_select: reg_select_type,
    context: binary_data
)

Derive_Rsp = (
    (rc_success,
    ksym_data) /
    rc_reg /
    rc_buffer /
    rc_command /
    rc_failure
)

DpDerive = (
    CC_DpDerive,
    reg_select: reg_select_type,
    context: nil / binary_data
)

DpDerive_Rsp = (
```



```
    rc_success /
    rc_reg /
    rc_buffer /
    rc_command /
    rc_failure
)

PublicRead = (
    CC_PublicRead,
    retricted: bool,
    context: binary_data,
)

PublicRead_Rsp = (
    (rc_success,
    kpub_data) /
    rc_buffer /
    rc_command /
    rc_failure
)

Quote = (
    CC_Quote,
    reg_select: reg_select_type,
    nonce: digest_data,
    context: binary_data
)

Quote_Rsp = (
    (rc_success,
    sign_data) /
    rc_reg /
    rc_buffer /
    rc_command /
    rc_failure
)

Sign = (
    CC_Sign,
    context: binary_data,
    digest: digest_data,
)

Sign_Rsp = (
    (rc_success,
    sign_data) /
    rc_buffer /
    rc_command /
```

```
    rc_failure
)

SignatureVerify = (
    CC_SignatureVerify,
    retricted: bool,
    context: binary_data,
    digest: digest_data,
    signature: sign_data,
)

SignatureVerify_Rsp = (
    (rc_success,
    bool) /
    rc_buffer /
    rc_command /
    rc_failure
)

; MARS Command
mars_command = [
        SelfTest /
        CapabilityGet /
        SequenceHash /
        SequenceUpdate /
        SequenceComplete /
        PcrExtend /
        RegRead /
        Derive /
        DpDerive /
        PublicRead /
        Quote /
        Sign /
        SignatureVerify
]

; MARS Response
; Due to pattern matching processing, the order of the response codes
; is important.  The most specific response code must be first.
; This type is for convenience of validation. Implementation should
; use the specific response types bound to the command.
mars_response = [
    rc:
        PublicRead_Rsp /
        Quote_Rsp /
        Sign_Rsp /
        RegRead_Rsp /
        SignatureVerify_Rsp /
```



```
        Derive_Rsp /
        CapabilityGet_Rsp /
        SequenceUpdate_Rsp /
        SequenceComplete_Rsp /
        SelfTest_Rsp /
        SequenceHash_Rsp /
        PcrExtend_Rsp /
        DpDerive_Rsp
]
```


Start of informative comment

Appendix B CBOR-encoded MARS Command and Response Examples

The tables in this appendix enumerate the hexadecimal values of examples of the MARS commands stated in the tables' titles, and the responses to those commands. These examples are based on a MARS using SHA-256 based algorithms as implemented in hw_sha2.c [4] and with a PS provisioned as "Here are thirty two secret bytes". The first byte in each cell of the VALUE column is defined in the CBOR Jump Table [2].

Table 1 - MARS_SelfTest

NAME	VALUE	DESCRIPTION
Array	82	2 data items
Command Code	00	MARS_CC_SelfTest
fullTest	f5	true
	RESPONSE	
Array	81	1 data item
Response Code	00	MARS_RC_SUCCESS

Table 2 - MARS_CapabilityGet

NAME	VALUE	DESCRIPTION
Array	82	2 data items
Command Code	01	MARS_CC_CapabilityGet
Property Tag	03	MARS_PT_LEN_DIGEST
	RESPONSE	
Array	82	2 data items
Response Code	00	MARS_RC_SUCCESS
Capability	1820	PROFILE_LEN_DIGEST

Table 3 - MARS_SequenceHash

NAME	VALUE	DESCRIPTION
Array	81	1 data item
Command Code	02	MARS_CC_SequenceHash
	RESPONSE	
Array	81	1 data item
Response Code	00	MARS_RC_SUCCESS

Table 4 - MARS_SequenceUpdate

NAME	VALUE	DESCRIPTION
Array	82	2 data items
Command Code	03	MARS_CC_SequenceUpdate
Input	4d 544347204d415253 2064656d6f	13-byte string "TCG MARS demo"
RESPONSE		
Array	82	2 data items
Response Code	00	MARS_RC_SUCCESS
Output	40	0-byte string

Table 5 - MARS_SequenceComplete

NAME	VALUE	DESCRIPTION
Array	81	1 data item
Command Code	04	MARS_CC_SequenceComplete
RESPONSE		
Array	82	2 data items
Response Code	00	MARS_RC_SUCCESS
Output	5820 cf5fb1917db493fd cd89e406fd47195c f51c82079dee5681 edd172cea2db819a	32-byte string digest

Table 6 - MARS_PcrExtend

NAME	VALUE	DESCRIPTION
Array	83	3 data items
Command Code	05	MARS_CC_PcrExtend
pcrIndex	00	PCR0
digest	5820 cf5fb1917db493fd cd89e406fd47195c f51c82079dee5681 edd172cea2db819a	32-byte string digest
RESPONSE		
Array	81	1 data item
Response Code	00	MARS_RC_SUCCESS

Table 7 - MARS_RegRead

NAME	VALUE	DESCRIPTION
Array	82	2 data items
Command Code	06	MARS_CC_RegRead
pcrIndex	00	PCR0
RESPONSE		
Array	82	2 data items
Response Code	00	MARS_RC_SUCCESS
Output	5820 633edbbf32fddb11 33ccf024c28e23a4 37d055d38dae8314 897be55c8c993a74	32-byte string PCR0's contents

Table 8 - MARS_Derive

NAME	VALUE	DESCRIPTION
Array	83	3 data items
Command Code	07	MARS_CC_Derive
regSelect	01	bitmask, selects PCR0
context	50 5365616c65645374 6f726167654b6579	16-byte string "SealedStorageKey"
RESPONSE		
Array	82	2 data items
Response Code	00	MARS_RC_SUCCESS
Output	5820 f16545d50164ad2c d4d4434f9e786a61 396bd9b49666c924 14cb0a78c8a5bc20	32-byte string derived value

Table 9 - MARS_DpDerive

NAME	VALUE	DESCRIPTION
Array	83	3 data items
Command Code	08	MARS_CC_DpDerive
regSelect	01	bitmask, selects PCR0
context	45 6368696c64	5-byte string, "child"

RESPONSE		
Array	81	1 data item
Response Code	00	MARS_RC_SUCCESS

Table 10 - MARS_PublicRead

NAME	VALUE	DESCRIPTION
Array	83	3 data items
Command Code	09	MARS_CC_PublicRead
restricted	f5	true
context	43 414b31	3-byte string, “AK1”
RESPONSE		
Array	81	1 data item
Response Code	05	MARS_RC_COMMAND

Table 11 - MARS_Quote

NAME	VALUE	DESCRIPTION
Array	84	4 data items
Command Code	0a	MARS_CC_Quote
regSelect	01	bitmask, selects PCR0
nonce	5820 48984ce5d39b6e27 1e91bfaadaa15baf ccfd32d8e192b9ea 5dfc6f0aa3997201	32-byte string number used once
context	43 414b31	3-byte string, “AK1”
RESPONSE		
Array	82	2 data items
Response Code	00	MARS_RC_SUCCESS
signature	5820 630b4e485c3013ef 57c27666383f8e2b ab517e2dedb05c37 6d91cd3075635ce2	32-byte string

Table 12 - MARS_Sign

NAME	VALUE	DESCRIPTION
Array	83	3 data items
Command Code	0b	MARS_CC_Sign

context	44 756b6579	4-byte string, "ukey"
digest	5820 48984ce5d39b6e27 1e91bfaadaa15baf ccfd32d8e192b9ea 5dfc6f0aa3997201	32-byte string nonce
RESPONSE		
Array	82	2 data items
Response Code	00	MARS_RC_SUCCESS
signature	5820 fd13be51c5193c3d 5fe0fad67439714e c62fb2abd9702e48 eec1312dd5083c3b	32-byte string

Table 13 - MARS_SignatureVerify

NAME	VALUE	DESCRIPTION
Array	85	5 data items
Command Code	0c	MARS_CC_SignatureVerify
restricted	f5	true
context	43 414b31	3-byte string, "AK1"
digest	5820 883e3e6b7f7c00f9 d23c4d0a3aa8d890 db348f03b3fa5a06 919d2ca2b6c609f8	32-byte string snapshot*
signature	5820 630b4e485c3013ef 57c27666383f8e2b ab517e2dedb05c37 6d91cd3075635ce2	32-byte string
RESPONSE		
Array	82	2 data items
Response Code	00	MARS_RC_SUCCESS
result	f5	true

* Note: when verifying a MARS signature, as in this example, the snapshot is computed by duplicating the functionality of CryptSnapshot() defined in [5].

End of informative comment

