Credential Management

authenticatorCredentialManagement

This command is used by the platform to manage resident credentials on the authenticator.

It takes the following input parameters:

Parameter name	Data type	Definition
subCommand (0x01)	Unsigned Integer	subCommand currently being requested
subCommandParams (0x02)	CBOR Map	Map of subCommands parameters.
pinProtocol (0x03)	Unsigned Integer	PIN protocol version chosen by the client.
pinAuth (0x04)	Byte Array	First 16 bytes of HMAC-SHA-256 of contents using pinToken.

The list of sub commands for credential management is:

subCommand Name	subCommand Number
getCredsMetadata	0x01
enumerateRPsBegin	0x02
enumerateRPsGetNextRP	0x03
enumerateCredentialsBegin	0x04
enumerateCredentialsGetNextCredential	0x05
deleteCredential	0x06

subCommandParams Fields:

Field name	Data type	Definition
rpIDHash (0x01)	Byte Array	RPID SHA-256 hash
credentialID (0x02)	PublicKeyCredentialDescriptor	Credential Identifier

On success, authenticator returns the following structure in its response:

Parameter name	Data type	Definition
existingResidentCredentialsCount (0x01)	Unsigned Integer	Number of existing resident credentials present on the authenticator.
maxPossibleRemainingResidentCredentialsCount	Unsigned Integer	Number of maximum possible

(0x02)		remaining resident credentials which can be created on the authenticator.
rp (0x03)	PublicKeyCredentialRpEntity	RP Information
rpIDHash (0x04)	Byte Array	RPID SHA-256 hash
totalRPs (0x05)	Unsigned Integer	total number of RPs present on the authenticator
user (0x06)	PublicKeyCredentialUserEntity	User Information
credentialID (0x07)	PublicKeyCredentialDescriptor	PublicKeyCredentialDescriptor
publicKey (0x08)	COSE_Key	Public key of the credential
totalCredentials (0x09)	Unsigned Integer	Total number of credentials present on the authenticator for the RP in question
credProtect (0x0A)	Unsigned Integer	Credential protection policy

Feature detection

To detect whether authenticator supports this preview feature, following conditions MUST be met:

- Authenticator MUST return "FIDO_2_1_PRE" in authenticatorGetInfo as one of version it supports in addition to "FIDO_2_0".
- Authenticator MUST return "credentialMgmtPreview" in options fields of authenticatorGetInfo and it MUST be set to true.
- For this preview feature, authenticatorUserVerification command is choosen from vendor command space and its value is MUST be 0x41.

Getting Credentials Metadata

Following operations are performed to get credentials metadata information :

- Platform sends authenticatorCredentialManagement command with following parameters:
    - subCommand (0x01): getCredsMetadata (0x01).
    - pinProtocol (0x03): Pin Protocol used. Currently this is 0x01.
    - pinAuth (0x04): `LEFT(HMAC-SHA-256(pinToken, getCredsMetadata (0x01)), 16)`.
- Authenticator verifies pinAuth by generating `LEFT(HMAC-SHA-256(pinToken, getCredsMetadata (0x01)), 16)` and matching against input pinAuth parameter.
    - If pinAuth verification fails, authenticator returns CTAP2_ERR_PIN_AUTH_INVALID error.
    - If authenticator sees 3 consecutive mismatches, it returns CTAP2_ERR_PIN_AUTH_BLOCKED indicating that power recycle is needed for further operations. This is done so that malware running on the platform should not be

able to block the device without user interaction.

- Authenticator returns authenticatorCredentialManagement response with following parameters:
    - existingResidentCredentialsCount (0x01) : total number of resident credentials existing on the authenticator.
    - maxPossibleRemainingResidentCredentialsCount (0x02) : maximum number of possbile remaining credentials that can be created on the authetenticator. Note that this number is an estimate as actual space consumed to create a credential depends on various conditions like which algorithm is picked, user entity information etc.

Enumerating RPs

Following operations are performed to enumerate RPs present on the authenticator:

- Platform gets pinToken from the authenticator.
- Platform sends authenticatorCredentialManagement command with following parameters:
    - subCommand (0x01): enumerateRPsBegin (0x02).
    - pinProtocol (0x03): Pin Protocol used. Currently this is 0x01.
    - pinAuth (0x04): `LEFT(HMAC-SHA-256(pinToken, enumerateRPsBegin (0x02)), 16)`.
- Authenticator verifies pinAuth by generating `LEFT(HMAC-SHA-256(pinToken, enumerateRPsBegin (0x02)), 16)` and matching against input pinAuth parameter.
    - If pinAuth verification fails, authenticator returns CTAP2_ERR_PIN_AUTH_INVALID error.
    - If authenticator sees 3 consecutive mismatches, it returns CTAP2_ERR_PIN_AUTH_BLOCKED indicating that power recycle is needed for further operations. This is done so that malware running on the platform should not be able to block the device without user interaction.
- Authenticator returns authenticatorCredentialManagement response with following parameters:
    - rp (0x03): PublicKeyCredentialRpEntity
    - rpIDHash (0x04) : RP ID SHA-256 hash.
    - totalRPs (0x05) : Total number of RPs present on the authenticator.
- Platform on receiving more than 1 totalRPs, performs following procedure for (totalRPs - 1) number of times:
    - Platform sends authenticatorCredentialManagement command with following parameters:
        - subCommand (0x01): enumerateRPsGetNextRP (0x03).
    - Authenticator on receiving such enumerateCredentialsGetNext subCommand returns authenticatorCredentialManagement response with following parameters:
        - rp (0x03): PublicKeyCredentialRpEntity
        - rpIDHash (0x04) : RP ID SHA-256 hash.

Enumerating Credentials for an RP

Following operations are performed to enumerate credentials for an RP:

- Platform gets pinToken from the authenticator.

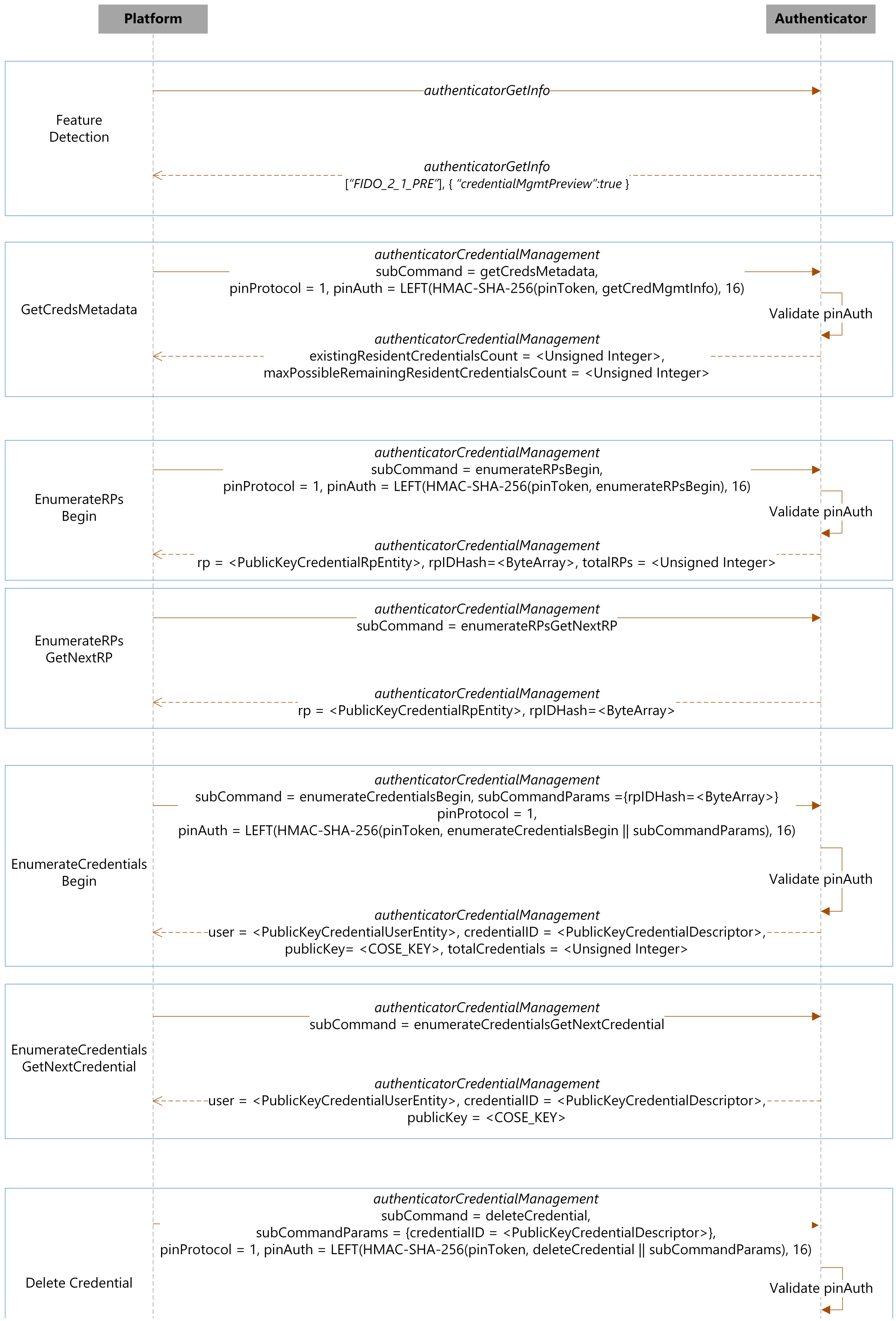
- Platform sends authenticatorCredentialManagement command with following parameters:
    - subCommand (0x01): enumerateCredentialsBegin (0x04).
    - subCommandParams (0x03): Map containing following parameters
        - rpIDHash (0x01): RPID SHA-256 hash.
    - pinProtocol (0x03): Pin Protocol used. Currently this is 0x01.
    - pinAuth (0x04): `LEFT(HMAC-SHA-256(pinToken, enumerateCredentialsBegin (0x04) || subCommandParams), 16)`.
- Authenticator verifies pinAuth by generating `LEFT(HMAC-SHA-256(pinToken, enumerateCredentialsBegin (0x04) || subCommandParams), 16)` and matching against input pinAuth parameter.
    - If pinAuth verification fails, authenticator returns CTAP2_ERR_PIN_AUTH_INVALID error.
    - If authenticator sees 3 consecutive mismatches, it returns CTAP2_ERR_PIN_AUTH_BLOCKED indicating that power recycle is needed for further operations. This is done so that malware running on the platform should not be able to block the device without user interaction.
    - If no credentials were found for this RPID hash, authenticator returns CTAP2_ERR_NO_CREDENTIALS.
- Authenticator returns authenticatorCredentialManagement response with following parameters:
    - user (0x06): PublicKeyCredentialUserEntity
    - credentialID (0x07): PublicKeyCredentialDescriptor
    - publicKey (0x08): public key of the credential in COSE_Key format
    - totalCredentials (0x09): total number of credentials for this RP
    - credProtect (0x0A): credential protection policy
- Platform on receiving more than 1 totalCredentials, performs following procedure for (totalCredentials - 1) number of times:
    - Platform sends authenticatorCredentialManagement command with following parameters:
        - subCommand (0x01): enumerateCredentialsGetNextCredential (0x05).
    - Authenticator on receiving such enumerateCredentialsGetNext subCommand returns with following parameters:
        - user (0x06): PublicKeyCredentialUserEntity
        - credentialID (0x07): PublicKeyCredentialDescriptor
        - publicKey (0x08): public key of the credential in COSE_Key format
        - credProtect (0x0A): credential protection policy

DeleteCredential

Following operations are performed to delete a credential:

- Platform gets pinToken from the authenticator.
- Platform sends authenticatorCredentialManagement command with following parameters:
    - subCommand (0x01): deleteCredential (0x06).

  - subCommandParams (0x02): Map containing following parameters
    - credentialsId (0x02): PublicKeyCredentialDescriptor of the credential to be deleted.
  - pinProtocol (0x03): Pin Protocol used. Currently this is 0x01.
  - pinAuth (0x04): `LEFT(HMAC-SHA-256(pinToken, deleteCredential (0x06) || subCommandParams), 16)`.
- Authenticator verifies pinAuth by generating `LEFT(HMAC-SHA-256(pinToken, deleteCredential (0x03) || subCommandParams), 16)` and matching against input pinAuth parameter.
  - If pinAuth verification fails, authenticator returns CTAP2_ERR_PIN_AUTH_INVALID error.
  - If authenticator sees 3 consecutive mismatches, it returns CTAP2_ERR_PIN_AUTH_BLOCKED indicating that power recycle is needed for further operations. This is done so that malware running on the platform should not be able to block the device without user interaction.
  - If there are not credential existing matching credentialDescriptor, return CTAP2_ERR_NO_CREDENTIALS.
  - Delete the credential and return CTAP2_OK.


Platform
Authenticator
Feature Detection
authenticatorGetInfo
authenticatorGetInfo
["FIDO_2_1_PRE"], { "credentialMgmtPreview":true }
GetCredsMetadata
authenticatorCredentialManagement
subCommand = getCredsMetadata,
pinProtocol = 1, pinAuth = LEFT(HMAC-SHA-256(pinToken, getCredMgmtInfo), 16)
Validate pinAuth
authenticatorCredentialManagement
existingResidentCredentialsCount = <Unsigned Integer>,
maxPossibleRemainingResidentCredentialsCount = <Unsigned Integer>
EnumerateRPs Begin
authenticatorCredentialManagement
subCommand = enumerateRPsBegin,
pinProtocol = 1, pinAuth = LEFT(HMAC-SHA-256(pinToken, enumerateRPsBegin), 16)
Validate pinAuth
authenticatorCredentialManagement
rp = <PublicKeyCredentialRpEntity>, rpIDHash=<ByteArray>, totalRPs = <Unsigned Integer>
EnumerateRPs GetNextRP
authenticatorCredentialManagement
subCommand = enumerateRPsGetNextRP
authenticatorCredentialManagement
rp = <PublicKeyCredentialRpEntity>, rpIDHash=<ByteArray>
EnumerateCredentials Begin
authenticatorCredentialManagement
subCommand = enumerateCredentialsBegin, subCommandParams ={rpIDHash=<ByteArray>}
pinProtocol = 1,
pinAuth = LEFT(HMAC-SHA-256(pinToken, enumerateCredentialsBegin || subCommandParams), 16)
Validate pinAuth
authenticatorCredentialManagement
user = <PublicKeyCredentialUserEntity>, credentialID = <PublicKeyCredentialDescriptor>,
publicKey= <COSE_KEY>, totalCredentials = <Unsigned Integer>
EnumerateCredentials GetNextCredential
authenticatorCredentialManagement
subCommand = enumerateCredentialsGetNextCredential
authenticatorCredentialManagement
user = <PublicKeyCredentialUserEntity>, credentialID = <PublicKeyCredentialDescriptor>,
publicKey = <COSE_KEY>
Delete Credential
authenticatorCredentialManagement
subCommand = deleteCredential,
subCommandParams = {credentialID = <PublicKeyCredentialDescriptor>},
pinProtocol = 1, pinAuth = LEFT(HMAC-SHA-256(pinToken, deleteCredential || subCommandParams), 16)
Validate pinAuth

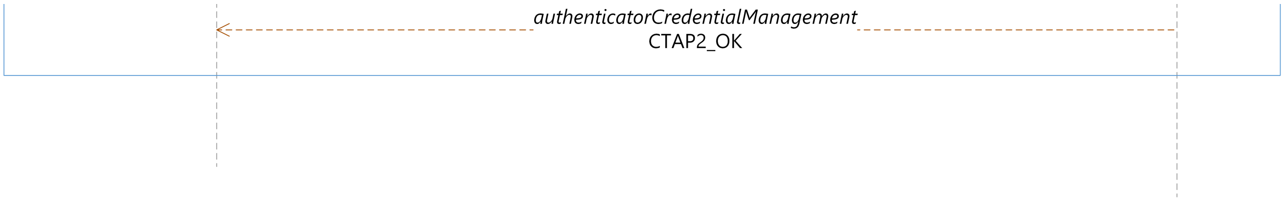


Figure 1 *Credential Management*

§ 1.1. Commands

For each command that contains parameters, the parameter map keys and value types are specified below:

Command	Parameter Name	Key	Value type
authenticatorCredentialManagement	subCommand	0x01	Unsigned Integer. (CBOR major type 0)
	subCommandParams	0x02	CBOR definite length map (CBOR major type 5)
	pinProtocol	0x03	Unsigned Integer. (CBOR major type 0)
	pinAuth	0x04	byte string (CBOR major type 2).

§ 1.2. Responses

Response Message	Member Name	Key	Value type
authenticatorCredentialManagement_Response	existingResidentCredentialsCount	0x01	Unsigned integer (CBOR major type 0).
	maxPossibleRemainingResidentCredentialsCount	0x02	Unsigned integer (CBOR major type 0).
	rp	0x03	CBOR definite length map (CBOR major type 5).
	rpIDHash	0x04	byte string (CBOR

			major type 2).
	totalRPs	0x05	Unsigned integer (CBOR major type 0).
	user	0x06	CBOR definite length map (CBOR major type 5).
	credentialID	0x07	CBOR definite length map (CBOR major type 5).
	publickKey	0x08	CBOR definite length map (CBOR major type 5). COSE_Key
	totalCredentials	0x09	Unsigned integer (CBOR major type 0).
	credProtect	0x0A	Unsigned integer (CBOR major type 0).