Aliro Specification
Version 1.0

Document:	26-42802-001_Aliro_1.0_specification.pdf February 18, 2026
Sponsored by:	Connectivity Standards Alliance
Accepted by:	This document has been accepted for release by the Connectivity Standards Alliance Board of Directors on **February 18, 2026**
Abstract:	This specification describes data exchange protocol and access model between User Device and Reader. The defined protocol is based on asymmetric key model and can be used by Readers in a wide variety of environments and use cases. These include, but are not limited to, Home, Hospitality and Corporates.

508 Second Street, Suite 206 Davis, CA 95616 - USA

www.csa-iot.org

This page is intentionally blank

Connectivity Standards Alliance – Copyright Notice, License and Disclaimer

This page is intentionally blank

Revision history

Revision	Date	Details
1.0	February 18, 2026	Initial release

This page is intentionally blank

Table of Contents

List of Figures

This page is intentionally blank

List of Tables

This page is intentionally blank

1 Introduction

1.1 Scope

The scope of this document is the NFC and Bluetooth LE interface between a Reader and User Device to provide the Reader with the necessary information to make an access decision. The primary features include a mutual authentication protocol and access document structure and transmission protocol. Other interfaces such as between Reader and Access Manager or between User Device and Credential Issuer are out of scope.

1.2 Purpose

1.3 Conformance levels

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", “SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED”, "MAY", and "OPTIONAL" in this document are to be interpreted as described in [11],[13] when, and only when, they appear in all capitals, as shown here.

1.4 Conventions

1.4.1 Number formats

In this specification hexadecimal numbers are prefixed with the designation “0x” and binary numbers are prefixed with the designation “0b”. All other numbers are assumed to be decimal unless indicated otherwise within the associated text.

Binary numbers are specified as successive groups of 4 bits, separated by a space (“ “) character from the most significant bit (next to the 0b prefix and left most on the page) to the least significant bit (rightmost on the page), e.g. the binary number 0b0000 1111 represents the decimal number 15. Where individual bits are indicated (e.g. bit 3) the bit numbers are relative to the least significant bit (i.e. bit 0).

When a bit is specified as having a value of either 0 or 1 it is specified with an “x”, e.g. “0b0000 0xxx” indicates that the lower 3 bits can take any value but the upper 5 bits SHALL each be set to 0.

Fields in parenthesis (..) are optional or conditional.

The concatenation operation is represented by ||.

ASCII values are represented by “quotes”.

All multi-octets fields defined in this specification SHALL be formatted, used and transferred with the most significant byte first order unless otherwise specified. The x and y components of ECC P256 public keys are used throughout the specification as input to various algorithms, when binary input is needed these unsigned integers SHALL be formatted as big endian multi-octets fields left-padded with zero-value bytes as needed to reach 32 bytes.

1.4.2 RFU bits and bytes

Bits and bytes which are declared in this specification as RFU SHALL be set to 0 by the transmitting entity and the receiving entity SHALL ignore the content of RFU bits / bytes, unless specified otherwise

1.4.3 Future extensibility

To allow future extensions of this specification:

- unknown DER-TLVs exchanged by the expedited-phase commands SHALL be accepted by the receiver, the content of such unknown DER-TLV SHALL NOT be interpreted by the receiver. A User Device or Reader SHALL not interpret unknown TLV tags at any point in any structure.
- Unknown Attribute IDs present in the Payload field of Aliro messages SHALL be accepted by the receiver, the content of such unknown Attribute IDs SHALL NOT be interpreted by the receiver.

1.5 Provisional Status Notification

This section exists to inform of the results implications of the validation process on technical specification. As per Connectivity Standards Alliance Policies & Procedures, the following items are marked as provisional based on the results of the Aliro Standard Validation Event resolution.

- [Optional] User Device Descriptor, as in Select Command Response Message, NFC 10.2.1.2
- [Optional] User Device Descriptor, as in the TLV for BLE, UWB in 11.7.3.7.1 Proprietary Information Attribute ID
- [Optional] Aliro Bluetooth-LE only expedited standard flow & thus the Bluetooth-LE only Aliro flow declared in section 11.1

2 References[1]

[1] The Bluetooth Core specification, version 4.2
[2] Digital Key Technical Specification Version 4.0.0
[3] AIS-31 A proposal for: Functionality classes for random number generators
[4] BSI TR-03111 Elliptic Curve Cryptography - Version 2.10
[5] FIPS186-5 Digital Signature Standard - February 2023
[6] ISO 18013-5:2021 Personal identification — ISO-compliant driving license - Mobile driving license (mDL) application
[7] ISO 7816-4 Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange – May 2020
[8] NIST SP 800-38D Recommendation for Block Cipher Modes of Operation - Galois-Counter Mode (GCM) and GMAC - November 2007
[9] NIST SP 800-90 Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June 2015
[10] RFC 2104 HMAC: Keyed-Hashing for Message Authentication
[11] RFC 2119 Key words for use in RFCs to Indicate Requirement Levels
[12] RFC 5280 PKIX Certificate and CRL Profile - May 2008
[13] RFC 8174 Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words
[14] RFC 9052 CBOR Object Signing and Encryption (COSE): Structures and Process
[15] RFC 8610 A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures
[16] Bluetooth LE GATT specification Supplement v8.
[17] NIST SP 800-56C Rev 2 – August 2020
[18] NIST SP 800-108r1 – August 2022
[19] ITU-T X.690 - August 2015
[20] RFC 5869 HMAC-based Extract and Expand Key Derivation Function – May 2010
[21] NIST Publication FIPS-197
[22] NFC Forum Digital Protocol Technical Specification – Version 2.3
[23] CSA Product Security Specification – Version 1.0
[24] NIST SP 800-56A rev 3, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography, April 2018
[25] NIST SP 800-186, Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters, February 2023
[26] Global Platform Secure Channel Protocol '11' – Amendment F Version 1.3
[27] RFC 8949 Concise Binary Object Representation (CBOR)

[1] The version and date information in these references was correct at the time this document was released.

[28] IEEE, Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)
[29] https://csa-iot.org/all-solutions/matter/

3 Definitions

Access Credential	A set of information that contains all data necessary to perform the access transaction, this includes the Access Credential key pair and an optional Access Document.
Access Data Element	Standardized structure to define access information.
Access Document	Issued by a Credential Issuer. Contains Access Data Elements and the Access Credential public key.
Access Manager	A manager used to determine whether access should be granted and actuate locking mechanisms. An Access Manager may be embedded in the Reader.
Central	A Bluetooth LE device that initiates a Bluetooth connection (see [1])
Credential Issuer	Issuer of the Access Credential.
Initiator	An entity that starts the UWB ranging packet exchange by sending a first UWB Poll packet (see [2]).
Peripheral	A Bluetooth LE device that accepts an incoming Bluetooth connection request after advertising (see [1])
Reader	A reader device to read an Access Credential from a User Device and optionally send Access Credential information to an Access Manager.
Reader System Issuer	Issuer of the Reader.
Responder	An entity that responds to a UWB Poll packet (see [2]).
Revocation Data Element	Standardized structure to define revocation information
Revocation Document	Issued by the Credential Issuer. Contains revocation data elements.
User Device	A portable device containing one or more access credentials E.g., card, fob, tag, key, mobile phone, smartwatch etc.

3.1 Keys and Data

This section defines some of the key and data elements used later in the document to help understand the flow diagrams.

cryptogram value calculated using Kpersistent over signaling bitmap and Access Credential metadata. It proves that the User Device is in possession of the same symmetric key as the Reader.

Kpersistent symmetric long-term key that is used to derive session keys. It is stored in non-volatile memory by the User Device and Reader.

ExpeditedSK derived symmetric key used to encrypt confidential commands and responses payloads for AUTH1 and EXCHANGE.

StepUpSK derived symmetric key used to protect the Step-up phase and command and response payloads for ENVELOPE, GET RESPONSE and EXCHANGE.

BleSK symmetric key used to protect UWB session setup commands sent over Bluetooth LE.

Kdh symmetric key derived during a Diffie-Hellman operation.

ePubK/ePrivK denotes ephemeral public and private keys.

PubK/PrivK denotes long term public and private keys.

transaction_identifier unique identifier of the transaction which is used on Reader and User Device side as an input to cryptographic operations.

reader_identifier unique identifier of a single Reader or a group of Readers concatenated with a unique identifier for each Reader. It is used by the User Device to select the correct Access Credential.

OUI and **CID** are used throughout the document for extension purposes. These OUI and CID values are IEEE issued values, for more information see [28].

4 Acronyms and abbreviations

AAD	Additional Authenticated Data
AC	Access Control
AES	Advanced Encryption Standard
AID	Application Identifier
APDU	Application Protocol Data Unit
BLE	Bluetooth Low Energy
CBOR	Concise Binary Object Representation
CDDL	Concise Data Definition Language
CID	Company ID
DO	Data Object
FCI	File Control Information
GATT	Generic Attribute Profile
IKM	Input Keying Material
KDF	Key Derivation Function
L2CAP	Logical Link Control and Adaptation Protocol
LSB	Least Significant Byte
MSB	Most Significant Byte
NFC	Near Field Communication
NVM	Non Volatile Memory
OEM	Original Equipment Manufacturer
OOB	Out-of-Band
OUI	Organizationally Unique Identifier
PHY	Physical Layer
PSM	Protocol Service Multiplexer
RAN	Ranging Area Network
RFU	Reserved for Future Use
RKE	Remote Keyless Entry
SHA	Secure Hash Algorithm
SP0	STS Packets type 0 (packets with payload and no STS)
SPSM	Simplified Protocol / Service Multiplexer
STS	Scrambled Timestamp Sequence
SYNC	Synchronization Header

TLV	Tag Length Value
TTL	Time to Live
URSK	UWB Ranging Secret Key
UWB	Ultra-Wideband

5 Architecture

5.1 Overview

This specification consists of 4 different parts that together allow the Reader to make an access decision:

- An access protocol for the User Device and Reader to authenticate each other and securely transmit an Access Document.
- Transport protocols to transmit the Access Protocol messages.
- A trust framework to provide the User Device and Reader with the necessary information to verify and authenticate the received messages.
- An Access Document to provide the Reader with further information to inform the access decision.

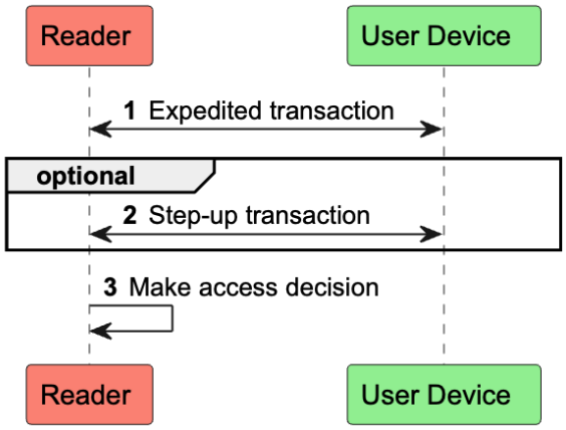
5.2 Access protocol

The access protocol used for the transaction proceeds in two phases as shown in Figure 5-1:

An **expedited** phase (see 8.1.1) in which the User Device proves to the Reader possession of a given secret key. If the Reader can make an access decision based on possession of this key, it can indicate success to the User Device and terminate the transaction.

A **document presentation** phase (see section 8.4), denoted as “step-up”, in which the User Device provides an Access Document, that is bound to the Access Credential public key, to the Reader which contains detailed information about the Access Credential that the Reader can use to make this access decision. If access is granted, the Reader MAY cache the Access Credential public key along with any associated access rights given in that document to ensure that subsequent transactions can be resolved during the expedited phase.

Figure 5-1 – High Level Transaction Flow



5.3 Transport protocols

A transaction can take place using either NFC or Bluetooth LE as the transport mechanism for the access protocol. When performing the transaction over Bluetooth LE, one of the flows uses UWB to securely determine proximity as specified in section 11.

5.4 Trust framework

This specification sets requirements for the keys and certificates used by the Credential Issuer, Reader System Issuer, Reader and User Device. It also defines a mechanism that can be used to transmit revocation information.

5.5 Access Document

The Access Document can be used to transmit information that is signed by the Credential Issuer from the User Device to the Reader. This information includes the Access Credential public key used in the expedited phase, and optionally includes access information like identifiers, schedules, and permissions.

6 Trust Framework

6.1 Implementation assumptions

In the trust framework Access Manager, Credential Issuer and Reader System Issuer are defined entities. Whether these are distinct entities or are the same entity for a particular access solution is out of scope of this specification.

6.2 Provisioning

The User Device and Reader provisioning operations including lifecycle management are out of scope of this document. The following assumptions have been made about the data present on the Reader and the User Device before a transaction can be performed.

The Reader has an identifier (reader_identifier), a key pair (reader_PubK/PrivK) and optionally a certificate (reader_Cert). To authenticate the User Device or use an Access Document to inform the access decision, an Access Manager is used. An Access Manager can be located within the Reader, or the Reader can contact an Access Manager remotely, for example through a panel or back-end. To authenticate the User Device, the Access Manager has to either be provisioned with a list of one or more Access Credential public keys or use an Access Document to determine the Access Credential public key. To authenticate the Access Document, the Access Manager has to either be provisioned with one or more Credential Issuer public keys (IssuerKey_PubK) or one or more Credential Issuer CA certificates.

The reader_identifier consists of a reader_group_identifier and a reader_group_sub_identifier. The reader_group_identifier is stored in the Reader during installation and is used by the User Device implementation to lookup the Access Credential and to lookup the reader_PubK or Reader System Issuer CA certificate to be used for the transaction. The reader_group_identifier is bound to reader_group_identifier_key, and SHALL refer to exactly one of the following options:

- The public key of the Reader System Issuer CA of the reader_Cert of that reader
- The public key of the reader key pair of that Reader

The reader_group_sub_identifier value is a 16-byte random value picked by the Reader during installation and provided on each transaction. This value enables lookup of already established persistent symmetric key (Kpersistent) on User Device side for installations where multiple Readers share the same reader_group_identifier.

The reader_group_identifier SHALL be chosen such that it ensures practical global uniqueness.

The User Device implementation SHALL allow at least 16 reader_group_identifier to be bound to the same Access Credential. Note that this means that the Access Credential is bound to zero, one or multiple Reader System Issuer CA certificates.

The User Device has one or more data structures called an Access Credential. This Access Credential contains all the information necessary to perform the access transaction. This includes a single unique Access Credential key pair, one or more Reader identifiers, optional Reader Certificate information, optional mailbox and optional Bluetooth LE setup information.

An Access Credential MAY be associated with an Access Document signed by a Credential Issuer attesting for the Access Credential's public key and the associated access rights. This Access Document can be presented during the transaction upon request from the Reader.

An Access Credential MAY be issued with a Revocation Document signed by the Credential Issuer providing revocation information to a Reader. The Revocation Document can be presented during the transaction upon request from the Reader.

The data in an Access Document and Revocation Document are contained in data elements. If the Reader requests an Access Document or Revocation Document, it has to be provisioned with a list of data element identifiers to use in the DeviceRequest structure, see section 8.4.1.

A User Device MAY have multiple Access Credentials that contain the same reader identifier. In the scenario where a User Device has multiple Access Credentials that support the same reader identifier, the User Device SHALL have a method that provides the user the ability to use all the Access Credentials that contain that reader identifier.

To authenticate the Reader and verify the Reader signature, the User Device has to either be provisioned with the Reader public key (reader_PubK) or with the Reader System Issuer CA certificate that issued the reader_Cert.

For further informative guidance on provisioning see CR-31968 (pending publication, available at https://groups.csa-iot.org/wg/aliro-tsg/document/31968).

6.3 Reader key pair

A Reader SHALL have a key pair (reader_PubK/reader_PrivK) and MAY have a certificate (reader_Cert) containing the Reader public key. The Reader uses this key pair to authenticate itself to the User Device.

A Reader SHALL support reader_PubK authentication method as defined in section 8.3.3.4.

A Reader key pair SHALL be ECC P-256 as specified in [5].

6.3.1 Reader certificate

The reader_Cert is an X.509 certificate issued by the Reader System Issuer CA and SHALL follow the format as defined in section 13.1.

A Reader SHALL support reader_Cert authentication method as defined in section 8.3.3.3.

A Reader SHALL accept a 3rd party reader_Cert certificate. Methods for loading reader_Cert into a Reader are out of scope of this specification.

If a reader_Cert is present on the Reader and it is configured for use, the Reader SHALL utilize the reader_Cert authentication method.

If reader_Cert is not present on the Reader or it is not configured for use, the Reader SHALL utilize the reader_PubK authentication method.

If the reader_Cert is present on the Reader, it SHALL be presented to the User Device using either LOAD CERT or AUTH1 command (see section 8.3.3.3 and section 8.3.3.4).

The reader_Cert SHALL be presented in compressed format as defined in section 13.3. The format in which the reader_Cert is stored on the Reader is out of scope of this specification.

The User Device SHALL accept at least 274 bytes of reader_Cert transmitted as compressed format when either the LOAD CERT (see section 8.3.3.3) or the AUTH1 command (see section 8.3.3.4) is sent.

If the User Device received the reader_Cert during the transaction, it SHALL use its stored Reader System Issuer CA certificate or Reader System Issuer CA public key to verify the reader_Cert presented at transaction time.

The User Device SHALL support verification of the reader_Cert using the Reader System Issuer CA public key.

The User Device uses the Reader System Issuer CA public key to verify the Reader certificate presented. If the reader_Cert is received, the following steps SHALL be performed:

1. First, the User Device SHALL verify the reader_Cert with the Reader System Issuer CA's public key. The User Device SHOULD verify the reader_Cert expiration time, the User Device SHALL verify all other elements of the reader_Cert.
2. Then the Reader public key (reader_PubK) extracted from reader_Cert SHALL be used by the User Device to authenticate the Reader.

If the User Device is unable to authenticate the Reader using the provided certificate and signature, an error status word SHALL be returned.

6.4 Access Credential key pair

A User Device SHALL have an Access Credential with a key pair (credential_PubK / credential_PrivK) that is unique to the User Device. If an Access Document is present, it SHALL contain the public key of the Access Credential.

To authenticate the User Device and verify the User Device signature, the Access Manager has to either know the Access Credential public key prior to the transaction or request and verify an Access Document.

An Access Credential key pair SHALL be ECC P-256 as specified in [5].

6.5 Credential Issuer key pair

An Access Document SHALL be signed by the Credential Issuer private key (IssuerKey_PrivK). Optionally the Credential Issuer public key (IssuerKey_PubK) is contained in the Credential Issuer Certificate (see section 6.5.1) issued by the Credential Issuer CA.

To authenticate the Access Document and verify the Access Document signature, the Reader has to know the IssuerKey_PubK. This can be done for example by storing IssuerKey_PubKs directly or by using the Credential Issuer CA certificate that issued the Credential Issuer Certificate (Issuer_Cert).

A Credential Issuer key pair SHALL be ECC P-256 as specified in [5].

6.5.1 Credential Issuer certificate

The Issuer_Cert is an X.509 certificate issued by the Credential Issuer CA and SHALL follow the format as defined in section 13.1.

The Reader uses the Credential Issuer CA public key to verify the Credential Issuer certificate (Issuer_Cert) presented.

If the Reader received the Issuer_Cert during the transaction and it does not yet trust the IssuerKey_PubK contained in it, it SHALL use its stored Credential Issuer CA certificate or Credential Issuer CA public key to verify the Issuer_Cert presented at transaction time. Then the

IssuerKey_PubK extracted from the Issuer_Cert SHALL be used by the Reader to verify the Access Document signature.

6.6 Revocation

The method used for Access Credential revocation depends on the installation type and Reader capabilities. When the Reader is connected to a backend system or panel, the revocation information does not need to be transferred using the transaction described in this document. Instead the Reader obtains or queries the revocation information from the backend or panel directly. This revocation method is out of the scope of this specification.

When the Reader is not connected to a backend or panel, the revocation information can be transported from a User Device to the Reader using the mechanisms described in section 7.6 and section 8.4.

7 Access Document

7.1 Overview

As part of the step-up phase (see section 5.2), an Access Document or Revocation Document (see section 7.6) can be securely transmitted from the User Device to the Reader. When a User Device presents an Access Document to a Reader, the Reader can use the information from the Access Document to help inform its access decision. This Access Document consists of data signed by a Credential Issuer. A Credential Issuer can vary greatly in the number of Access Credentials that it manages, ranging from a single Access Credential in a household or an Access Credential for each employee in a global enterprise. A Reader can be setup to trust credentials from one or multiple Credential Issuers.

After the Reader has determined that it trusts the Credential Issuer (see section 6) that has issued the Access Document, it will look at the data elements and the signatures within that Access Document. The data elements will provide information to the Reader to inform its access decision.

The Access Document consists of two parts. The first one is the IssuerAuth structure, that contains the cryptographic details to authenticate the data elements and Access Credential public key and to verify the validity of the IssuerAuth structure. The second are the data elements encapsulated in the IssuerSignedItem fields. An Access Document can contain one or more data elements (IssuerSignedItem) which can be requested individually by the Reader.

The IssuerAuth structure contains hashes of all the IssuerSignedItems (i.e. data elements) of the Access Document. This allows the User Device to only return the data elements that are requested, while allowing all the data elements that are returned to be validated using only a single signature from the Credential Issuer.

This specification defines two standardized data elements:

- the Access Data Element, that provides a standardized structure to define access information for use in the Access Document.
- the Revocation Data Element, that provides a standardized structure to define revocation information for use in the Revocation Document.

Section 8.4 describes how data elements are requested and transmitted.

The User Device SHALL support storage and retrieval of the Access Document.

The Reader MAY support retrieval and verification of the Access Document.

7.2 Access Document structure

The Access Document is based on mechanisms defined in [6]. The Access Document contains the IssuerAuth structure and IssuerSignedItem structures that SHALL be implemented according to section 9.1.2 of [6] except for the changes and additional requirements described in this section.

All CBOR encoded structures used in this specification SHALL be encoded according to the core deterministic requirements in section 4.2.1 of [27].

7.2.1 Cryptographic requirements

The digest algorithm SHALL be SHA-256 as defined in section 9.1.2 of [6].

The signature algorithm SHALL be “ES256” (ECDSA with SHA-256) with the P-256 curve as defined in section 9.1.2 of [6].

To select and trust the Credential Issuer key pair, the IssuerAuth structure SHALL contain at least one of the following two fields and MAY contain both. If both are present the Reader decides which fields to use.

- The x5chain field in the header of IssuerAuth as specified in section 9.1.2 of [6]. This field MAY be present. If present this field SHALL contain the Credential Issuer certificate as defined in 6.5.1.
- The key identifier (“kid”) header as defined in [14]. This field MAY be present, if present the key identifier SHALL be calculated by taking the first 8 bytes of SHA256(“key-identifier” || 0x04 || IssuerKey_PubK), where “key-identifier” is the literal string and IssuerKey_PubK takes the form of IssuerKey_PubK.x || IssuerKey_PubK.y for P-256.

Other fields MAY also be present in the header of the IssuerAuth structure.

7.2.2 Structure requirements

Note that when referring to CBOR maps, the terms ‘key’ and ‘value’ are used to describe the key-value pairs used in a map.

The structure that is signed as part of the IssuerAuth structure is the MobileSecurityObject. The following additional requirements apply:

The "keyAuthorizations" field SHALL NOT be present.

The “keyInfo” and “expectedUpdate” SHOULD NOT be present.

An additional field SHALL be present in the MobileSecurityObject with the key as “TimeVerificationRequired” and the boolean field TimeVerificationRequired. The meaning of this field is defined in section 7.2.4.

An additional field MAY be present in the “validityInfo” field in the MobileSecurityObject with the key as “validityIteration” and the integer field ValidityIteration. The meaning of this field is defined in section 7.2.3.

To optimize the size of the IssuerAuth structure the keys in the maps SHALL be replaced by a different key (note that these are integers, still encoded as text strings) according to Table 7-1. Some of the keys are nested, shown by indentations.

Table 7-1 – New IssuerAuth key values

Original key	New key
“version”	“1”
“digestAlgorithm”	“2”
“valueDigests”	“3”
“deviceKeyInfo”	“4”
“deviceKey”	“1”
“keyInfo”	“2”
“docType”	“5”
“validityInfo”	“6”
“signed”	“1”
“validFrom”	“2”
“validUntil”	“3”
“expectedUpdate”	“4”
“validityIteration”	“5”
“timeVerificationRequired”	“7”

To optimize the size of the IssuerSignedItem structure the keys in the maps SHALL be replaced by a different key (note that these are integers, still encoded as text strings) according to Table 7-2.

Table 7-2 – New IssuerSignedItem key values

Original key	New key
“digestID”	“1”
“random”	“2”
“elementIdentifier”	“3”
“elementValue”	“4”

7.2.3 Validity iteration

The ValidityIteration field MAY be present. It contains a validity iteration indicator that can be used by the Credential Issuer to invalidate older iterations of Access Credentials or Revocation Credentials. The Reader SHALL store two ValidityIteration values per Credential Issuer, called

AccessIteration and RevocationIteration. If the Reader does not know the AccessIteration for a Credential Issuer then the Reader SHALL set the AccessIteration to 0. If the Reader does not know the RevocationIteration for a Credential Issuer, then the Reader SHALL set the RevocationIteration to 0.

When the Reader receives an Access Document with a ValidityIteration with a value greater than AccessIteration, it SHALL update the value of AccessIteration with the value of ValidityIteration and SHALL invalidate all stored Kpersistent keys associated with that Credential Issuer and with a difference between ValidityIteration and AccessIteration greater than 8.

When the ValidityIteration field in the Access Document is less than AccessIteration and the difference is equal to or greater than 8, the Access Document is not valid.

When the ValidityIteration field in the Access Document is less than AccessIteration and the difference is less than 8, the Access Document is valid.

When the ValidityIteration field in the Access Document is equal to or greater than AccessIteration, the Access Document is valid.

When the ValidityIteration field in the Revocation Document is less than RevocationIteration, then the Revocation Document is invalid.

When the ValidityIteration filed in the Revocation Document is equal to or greater than the RevocationIteration, then the Revocation Document is valid.

7.2.4 Time verification

The TimeVerificationRequired field SHALL be present in the IssuerAuth structure. The time policy is used by the Credential Issuer to define the behavior of a Reader when verifying time-based elements.

When verifying a time-based element, the Reader SHALL follow the following requirements:

If the Reader can validate the time-based element the Reader SHALL validate the time-based element.

If the Reader is not able to validate the time-based element and TimeVerificationRequired is set to True the Reader SHALL consider the verification to be invalid.

If the Reader is not able to validate the time-based element and the TimeVerificationRequired is set to False the Reader MAY consider the element invalid or MAY consider the element valid.

7.2.5 DocType, NameSpace and Data Element Identifier

The Access Document and Revocation Document each have their document type.

Each data element belongs to a specific namespace. Within a namespace, the data element is identified using the Data Element Identifier.

The DocType, NameSpace and DataElementIdentifier concepts are defined in [6] section 7.1.

The DocType and Namespace for the Access Document and Revocation Document are defined in 7.7.

7.3 Access Data Element

The Access Data Element is a CBOR (see [15]) encoded data element, following the requirements in section 8.1 of [6]. Section 8.1 of [6] uses CDDL (see [15]) to define CBOR structures. CDDL is also used for that purpose in this specification.

Note that section 7.2 requires all CBOR structures to be encoded according to the core deterministic requirements in section 4.2.1 of [27].

The Access Data Element consists of the identifier and the value of the data element. The identifier (see DataElementIdentifier as defined in [6]) is a string that SHALL have a maximum length of 128 characters. The data element identifier is used by the Credential Issuer to identify the access level indicated for that access data element. Examples of such values are “administrator”, “floor1”, “building2.front_door”.

By asking for a specific data element identifier or set of data element identifiers and verifying the data element identifiers of the data elements that it receives from the User Device, the Reader can verify the received data elements were intended for that Reader.

An Access Document MAY contain more than one Access Data Element and the Reader can request more than one Access Data Element identifier in a single request.

An Access Data Element SHALL have the following field:

- Version

An Access Data Element MAY have the following fields:

- ID
- Access rules
- Schedules
- Reader rules
- Access extensions
- Non-access extensions

The Reader SHALL support version and access rules in the Access Data Element. Support for schedules, access extensions, non-access extensions, reader rules and ID is OPTIONAL for a Reader.

Note that while this requirement means that support of the content for certain structures is optional, the processing rules as defined in 7.5 always apply. This is especially relevant for the requirements of rejecting the whole Access Data Element when encountering certain unknown structures like an unknown access extension.

The Access Data Element SHALL be formatted according to the following CDDL:

```
AccessData = {
    AccessData_Version => uint,
    ? AccessData_ID => bstr .size (1..16),
    ? AccessData_AccessRules => [1*8 AccessRule],
    ? AccessData_Schedules => [1*8 Schedule],
    ? AccessData_ReaderRuleIds => [1*8 uint .size 2],
    ? AccessData_NonAccessExtensions => {+ Vendor_RegisteredID => [+ NonAccessExtension] },
    ? AccessData_AccessExtensions => {+ Vendor_RegisteredID => [+ AccessExtension] }
}

Vendor_RegisteredID = uint .size 3
```

The values for the keys in the maps SHALL be:

```
AccessData_Version = 0
AccessData_ID = 1
AccessData_AccessRules = 2
AccessData_Schedules = 3
AccessData_ReaderRuleIds = 4
AccessData_NonAccessExtensions = 5
AccessData_AccessExtensions = 6
```

7.3.1 Access Data Element version

The AccessData_Version is used to indicate the version of the Access Data Element. The value SHALL be 1. A Reader SHALL verify if the version is supported, if the value is not supported, the Access Data Element is considered invalid.

7.3.2 ID

The AccessData_ID is a field that is used to indicate an ID. The content of the ID is out of scope of this specification. Example include a Credential ID, User ID or Device ID.

7.3.3 Access rules

The AccessRule structure can be used to limit the capabilities for which the Access Data Element is valid and/or the time-period for which the Access Data Element is valid.

If one or more AccessRule structures are present in an Access Data Element, the Reader SHALL verify if at least one of the AccessRule structures is valid. The Access Data Element is invalid if all the AccessRule structures are invalid.

Note that this means that an Access Data Element is valid if at least one of the AccessRule structures is valid. This also implies that it is possible for a Reader to stop parsing the other AccessRule structures once it has encountered an AccessRule structure that is valid.

The AccessRule_Capabilities contains the permissions that this AccessRule grants. The AccessRule_AllowScheduleIds and AccessRule_DenyScheduleIds contain a reference to schedules in the Schedule structure. Each referenced schedule in the AccessRule_AllowScheduleIds defines a period in which the AccessRule is valid. Each reference schedule in the AccessRule_DenyScheduleIds defines a period in which the AccessRule is not valid. The AccessRule_DenyScheduleIds takes precedence over the AccessRule_AllowScheduleIds.

An AccessRule structure is valid if none of the following checks are invalid. This SHALL be verified by the Reader.

- If the AccessRule_Capabilities field is present, the check results in invalid if the intended action by the Reader is not set to True in the AccessRule_Capabilities field.
- If the AccessRule_AllowScheduleIds is present, the check results in invalid if none of the referenced schedules is in-range.
- If the AccessRule_DenyScheduleIds is present, the check results in invalid if at least one of the reference schedules is in-range.

- If the AccessRule_AllowScheduleIds or AccessRule_DenyScheduleIds is present and the Reader cannot validate time-based fields and the TimeVerificationRequired field in the IssuerAuth structure is set to True, the check results in invalid.
- If the AccessRule_AllowScheduleIds or AccessRule_DenyScheduleIDs is present and the Reader does not support the Schedule structure, the check results in invalid.

The AccessRule structure SHALL be formatted according to the following CDDL:

```
AccessRule = {
    ? AccessRule_Capabilities => uint .bits AccessRuleCapabilitiesBits,
    ? AccessRule_AllowScheduleIds => uint .bits AccessRuleScheduleIdsBits,
    ? AccessRule_DenyScheduleIds => uint .bits AccessRuleScheduleIdsBits
}

AccessRuleCapabilitiesBits = &(
    Secure                      : 0,
    Unsecure                    : 1,
    Toggle_Secure_or_Unsecure   : 2,
    Momentary_Unsecure          : 3,
    Extended_Momentary_Unsecure : 4,
    Payment_Permission          : 5
) / (6..15) ; RFU

AccessRuleScheduleIdsBits = &(
    AccessDataSchedule1 : 0,
    AccessDataSchedule2 : 1,
    AccessDataSchedule3 : 2,
    AccessDataSchedule4 : 3,
    AccessDataSchedule5 : 4,
    AccessDataSchedule6 : 5,
    AccessDataSchedule7 : 6,
    AccessDataSchedule8 : 7,
)
```

The values for the keys in the maps SHALL be:

```
AccessRule_Capabilities = 0
AccessRule_AllowScheduleIds = 1
AccessRule_DenyScheduleIds = 2
```

The fields in the AccessRuleCapabilitiesBits as defined in the CDDL have the meaning as defined in Table 7-3.

Table 7-3 – Meaning of AccessRuleCapabilitiesBits

Field (bit)	Meaning
bit 0: Secure	Secure
bit 1: Unsecure	Unsecure
bit 2: Toggle_Secure_or_Unsecure	Toggle Secure or Unsecure
bit 3: Momentary_Unsecure	Momentary Unsecure
bit 4: Extended_Momentary_Unsecure	Extended Momentary Unsecure
bit 5: Payment_Permission	Payment Permission
bit 6..15 : RFU	RFU

The AccessRule structure SHALL contain at least one field. If a field is present, at least one of the bits SHALL be set to True.

7.3.4 Schedules

The AccessData_Schedules is a field that is used to define schedules. This field only defines the schedules, they are only applicable if referenced in AccessRule.

Support for the schedule structure is OPTIONAL for a Reader.

If the Reader supports the Schedule structure, it SHALL be able to parse and interpret all fields of the Schedule structure.

The Reader SHALL verify the schedule structure by determining whether the current time falls within the time defined by the schedule. If this is the case, the schedule is in-range.

The Schedule structure SHALL be formatted according to the following CDDL:

```
Schedule = {
    Schedule_StartPeriod => uint .size 4,
    ? Schedule_EndPeriod => uint .size 4,
    ? Schedule_RecurrenceRule => RecurrenceRule,
    Schedule_Flags => uint .bits Schedule_FlagsBits
}

RecurrenceRule = [
    RecurrenceRule_DurationSeconds : uint .size 4,
    RecurrenceRule_Mask : uint .bits RecurrenceRuleMaskBits / 0,
    RecurrenceRule_Pattern : RecurrenceRulePatternType,
    RecurrenceRule_Interval : uint .size 1,
    RecurrenceRule_Ordinal : RecurrenceRuleOrdinalValue,
]

Schedule_FlagsBits = &(
    Time_in_UTC: 0,
) / (1..7) ; RFU

RecurrenceRulePatternType = &(
    Daily             : 1,
    Weekly            : 2,
    MonthlyByWeekDay  : 3,
    MonthlyByDate     : 4,
    YearlyByWeekDay   : 5,
    YearlyByDate      : 6,
    YearlyByWeek      : 7,
```

```
    YearlyByMonthWeek : 8
)

RecurrenceRuleOrdinalValue = &(
    RecurrenceRuleOrdinal_Daily             : 0,
    RecurrenceRuleOrdinal_Weekly            : 0,
    RecurrenceRuleOrdinal_MonthlyByWeekday  : (-5..-1) / (1..5),
    RecurrenceRuleOrdinal_MonthlyByDate     : ((-31..-1) / (1..31)) / 0,
    RecurrenceRuleOrdinal_YearlyByWeekday   : (-5..-1) / (1..5),
    RecurrenceRuleOrdinal_YearlyByDate      : (-31..-1) / (1..31),
    RecurrenceRuleOrdinal_YearlyByWeek      : (-53..-1) / (1..53),
    RecurrenceRuleOrdinal_YearlyByMonthWeek : (-5..-1) / (1..5)
)

RecurrenceRuleMaskBits = &(
    RecurrenceRuleMask_Weekly           : RecurrenceRuleMaskBits_Weekdays,
    RecurrenceRuleMask_MonthlyByWeekDay : RecurrenceRuleMaskBits_Weekdays,
    RecurrenceRuleMask_MonthlyByDate    : RecurrenceRuleMaskBits_Dates /
RecurrenceRuleMaskBits_Weekdays,
    RecurrenceRuleMask_YearlyByWeekDay  : RecurrenceRuleMaskBits_Yearly,
    RecurrenceRuleMask_YearlyByDate     : RecurrenceRuleMaskBits_Yearly,
    RecurrenceRuleMask_YearlyByWeek     : RecurrenceRuleMaskBits_Yearly,
    RecurrenceRuleMask_YearlyByMonthWeek : RecurrenceRuleMaskBits_Yearly
)

RecurrenceRuleMaskBits_Weekdays = &(
    Monday    : 0,
    Tuesday   : 1,
    Wednesday : 2,
    Thursday  : 3,
    Friday    : 4,
    Saturday  : 5,
    Sunday    : 6
)

RecurrenceRuleMaskBits_Yearly = &(
    Monday    : 0,
    Tuesday   : 1,
    Wednesday : 2,
    Thursday  : 3,
    Friday    : 4,
    Saturday  : 5,
    Sunday    : 6,
    January   : 7,
    February  : 8,
    March     : 9,
    April     : 10,
    May       : 11,
    June      : 12,
    July      : 13,
    August    : 14,
    September : 15,
    October   : 16,
    November  : 17,
    December  : 18
)

RecurrenceRuleMaskBits_Dates = &(
    day1     : 0,
    day2     : 1,
    day3     : 2,
    day4     : 3,
    day5     : 4,
    day6     : 5,
    day7     : 6,
    day8     : 7,
    day9     : 8,
    day10    : 9,
    day11    : 10,
    day12    : 11,
    day13    : 12,
    day14    : 13,
```

```
    day15    : 14,
    day16    : 15,
    day17    : 16,
    day18    : 17,
    day19    : 18,
    day20    : 19,
    day21    : 20,
    day22    : 21,
    day23    : 22,
    day24    : 23,
    day25    : 24,
    day26    : 25,
    day27    : 26,
    day28    : 27,
    day29    : 28,
    day30    : 29,
    day31    : 30,
)
```

The values for the keys in the maps SHALL be:

```
Schedule_StartPeriod = 0
Schedule_EndPeriod = 1
Schedule_RecurrenceRule = 2
Schedule_Flags = 3
```

The Schedule_StartPeriod is a REQUIRED field that indicates the inclusive start time of the schedule in seconds since Unix epoch.

If Schedule_RecurrenceRule is not present, the Schedule_EndPeriod field SHALL be present, and it defines the non-inclusive end time of the schedule in seconds since Unix epoch.

If Schedule_RecurrenceRule is present, the Schedule_EndPeriod field MAY be present. If Schedule_EndPeriod is present, the event recurs until Schedule_EndPeriod. If Schedule_EndPeriod is not present the schedule recurs forever.

Schedule_RecurrenceRule is an OPTIONAL field. If present it defines how the event recurs.

The fields in ScheduleRecurrenceRule have the following definition:

RecurrenceRule_DurationSeconds defines the duration in seconds of each recurring event. The first event starts at Schedule_StartPeriod and lasts the duration with each subsequent event starting according to the other fields in the ScheduleRecurrenceRule structure.

RecurrenceRule_Pattern specifies the recurrence frequency and how to interpret the interval, ordinal and mask fields.

RecurrenceRule_Mask specifies the mask of when the event recurs. An event only recurs if the corresponding bit in the mask is set to True. The exact meaning depends on the pattern used:

- When the pattern is Daily, the mask is not used.
- When the pattern is Weekly or Monthly-by-Weekday, the mask is specified by RecurrenceRuleMaskBits_Weekdays.
- When the pattern is Monthly-by-Date, the mask is specified by RecurrenceRuleMaskBits_Dates or RecurrenceRuleMaskBits_Weekdays. See the RecurrenceRule_Ordinal definition for which value is used.
- When the pattern is Yearly-by-Weekday, Yearly-by-Date, Yearly-by-Week or Yearly-by-MonthWeek, the mask is specified by RecurrenceRuleMaskBits_Yearly.

RecurrenceRule_Interval specifies the recurrence interval. The exact meaning depends on the pattern used:

- When the Pattern is Daily, then the event recurs every Interval days.
- When the Pattern is Weekly, then the event recurs every Interval weeks.
- When the Pattern is Monthly, then the event recurs every Interval months.
- When the Pattern is Yearly, then the event recurs every Interval years.

The value specifies that the event recurs every xx days/weeks/months/years. Where xx is the value of the interval and days/weeks/months/years is specified by whether the pattern is daily, weekly, monthly or yearly.

RecurrenceRule_Ordinal specifies the ordinal of the recurring event, i.e. the event occurs on the nth weekday/week/month. The exact meaning depends on the pattern used:

- When the Pattern is Daily or Weekly, the Ordinal is not used.
- When the Pattern is Monthly-by-Weekday, then the Ordinal is the nth weekday of the month, where n is [-5..-1] or [1..5] and day is {M,T,W,Th,F,Sa,Su}.
- When the Pattern is Monthly-by-Date and the Ordinal is 0, the RecurrenceRule_Mask is RecurrenceRuleMaskBits_Dates. If the ordinal n is [-31..-1] or [1..31], the ordinal is the nth day of the month and the mask is used for RecurrenceRuleMaskBits_Weekdays.
- When the Pattern is Yearly-by-Weekday, then the Ordinal is the nth weekday of the month, where n is [-5..-1] or [1..5] and day is {M,T,W,Th,F,Sa,Su}.
- When the Pattern is Yearly-by-Date, then the Ordinal is the nth day of the month, where n is [-31..-1] or [1..31].
- When the Pattern is Yearly-by-Week, then the Ordinal is the nth week of the year, where n is [-53..-1] or [1..53].
- When the Pattern is Yearly-by-MonthWeek, then the Ordinal is the nth week of the month, where n is [-5..-1] or [1..5].

Schedule_Flags specifies whether the schedule requirements SHALL be interpreted according to the Reader local time or UTC time. When the Time_in_UTC bit is set, UTC time is used. When the Time_in_UTC bit is not set, local time is used.

7.3.5 Reader rules

The ReaderRuleIds structure is used to point to a set of Access Rules already present on the Reader, for example during provisioning of the Reader. These Access Rules structures referenced by the ReaderRuleIds SHALL use the same structure and SHALL be processed the same way as the Access Rule structures as defined for the Access Data Element.

When a Reader is not able to interpret a Reader Rule reference, the Access Data Element that contains the Reader Rule SHALL be considered invalid.

7.3.6 Non-access extensions

The non-access extension can be used to provide extra information to a Reader. The information in the non-access extension SHALL NOT contain information that modifies the access decision.

AccessData_Extension uses a Vendor_RegisteredID that SHALL be an IEEE OUI or CID.

A non-access extension SHALL NOT be used to provide information, functionality, or access right that can be described with an Access Data Element that does not use an access extension or non-access extension.

A Reader SHALL ignore any non-access extension that it is not able to interpret.

When a Reader can interpret a non-access extension, it SHALL NOT use the information in the extension to change the Access Decision. Any other behavior from the Reader is out of scope of this specification.

The NonAccessExtension structure SHALL be formatted according to the following CDDL:

```
NonAccessExtension = [
    Vendor_ExtensionID : uint,
    Version : uint,
    Data : bstr
]
```

The value of the `Vendor_ExtensionID`, `Version` and `Data` fields are out of scope of this specification.

7.3.7 Access extensions

The access extension can be used to provide extra information to a Reader. The information in the Access extension MAY contain information that modifies the access decision.

Access extensions SHALL NOT be used to provide information, functionality, or access rights that can be described with an Access Data Element that does not use an access extension or non-access extension.

AccessData_AccessExtensions uses a Vendor_RegisteredID that SHALL be an IEEE OUI or CID.

Each AccessExtension contains a Flag that SHALL be used to indicate whether the access extension is critical or non-critical. If the Flag is not set, the Reader SHALL consider access extension to be critical.

The intent of the criticality flag is that it can be used in situations where parsing but not interpreting an access extension can have unwanted consequences, this is done by requiring that a Reader either processes a critical access extension or rejects the Access Data Element that contains the critical access extension. Therefore, the following behavior for interpreting access extensions SHALL be implemented by all Readers and if an access extension is present, a Reader SHALL follow the following steps:

- If the critical flag is set to critical and the Reader does not interpret the access extension, the Reader SHALL ignore the Access Data Element and consider it invalid. The Reader SHALL NOT use this Access Data Element for access decision.
- If the critical flag is set to critical and the Reader interprets the access extension, how to determine the access decision is out of scope of this specification.
- If the critical flag is set to not critical and the Reader does not interpret the access extension, the Reader SHALL ignore the access extension. The Reader SHALL NOT use this access extension for access decision. The other structures in this Access Data Element SHALL be parsed and interpreted in accordance as normal.

- If the critical flag is set to not critical and the Reader interprets the access extension, how to determine the access decision is out of scope of this specification.

Note: the intention of “interpret” is that the Reader parses, understands and acts on the access extension.

A Reader SHALL NOT use information from an access extension to change its Access Decision if that information can be described with an Access Data Element that does not use an Extension or non-access extension.

The AccessExtension structure SHALL be formatted according to the following CDDL:

```
AccessExtension = [
    Criticality : uint .bits Criticality_Bits ,
    Vendor_ExtensionID : uint,
    Version : uint,
    Data : bstr
]

Criticality_Bits = &(
    Critical        : 0
) / (1..7) ; RFU
```

The value of the `Vendor_ExtensionID`, `Version` and `Data` fields are out of scope of this specification.

7.4 Access Document verification

Information from the Access Document SHALL NOT be used without validating the Access Document. When validating the Access Document, the inspection procedure for issuer data authentication as described in section 9.3.1 of [6] SHALL be replaced by the following, which SHALL be implemented by the Reader.

1. If the x5chain field is present, validate the end-entity certificate in the x5chain field.
2. Verify the digital signature of the IssuerAuth structure using the IssuerKey_PubK.
3. Calculate the digest value for every IssuerSignedItem returned in the device response structure according to section 9.1.2.5 of [6] and verify that these calculated digests equal the corresponding digest values in the IssuerAuth structure.
4. Verify that the DocType in the IssuerAuth structure matches the relevant DocType in the Documents structure in the DeviceResponse, see section 8.4.2.
5. The following time-based elements in the ValidityInfo structure SHALL be validated considering the TimeVerificationRequired element (see section 7.2.4):
    a. ‘validFrom’ <= ‘currentTime’ <= ‘validUntil’, meaning that: The current time SHALL be equal or later than the ‘validFrom’ field. The ‘validUntil’ SHALL be equal or later than the current time.
    b. The 'signed' date is within the validity period of the certificate in the Issuer_Cert (if applicable).
6. If present, a Reader SHALL verify the ValidityIteration field as defined in section 7.2.3.

If the Reader requests one or more data elements, but it receives no data elements, or all received data elements are considered invalid, it SHALL reject the Access Document.

When using an Access Document, a Reader SHOULD only use an Access Data Element to inform its Access Decision. An example of an exception to this recommendation is the application of a revocation list.

7.5 Access data element verification

When using an Access Document, a Reader SHALL NOT use any data element that is not an Access Data Element to inform its Access Decision.

A Reader SHALL NOT process any structure in the AccessData structure that is not specified in this specification.

A Reader SHALL NOT use any information from an Access Element without performing all the following processing steps:

- Process the Version field according to section 7.3.1.
- If present, process AccessRule structures according to section 7.3.3.
- If present, process the ReaderRuleId structure according to section 7.3.5.
- If present, process the AccessExtension structures according to section 7.3.7.

Information from an Access Data Element SHALL NOT be used if it is considered invalid.

A Reader MAY process non-access extensions. If it does, it SHALL process them according to section 7.3.6.

7.6 Revocation Document

This specification defines a Revocation Document that contains revocation information that the Reader can use to inform its access decision. The Revocation Document can be retrieved using the same mechanism used to retrieve the Access Document, see section 8.4.

Presentation of the Revocation Document is not bound to a particular User Device and therefore the Revocation Document structure SHALL NOT contain the DeviceKeyInfo element in the IssuerAuth structure.

The Revocation Document SHALL have the same structure, cryptographic and verification requirements as the Access Document with the following changes:

- The DocType SHALL be the doctype for the Revocation Document as defined in 7.7.
- The IssuerAuth structure SHALL NOT contain the deviceKeyInfo field.

The User Device SHALL support storage and retrieval of the Revocation Document.

The Reader MAY support retrieval and verification of the Revocation Document.

When receiving a Revocation Document, the Reader SHALL verify integrity, construction, and contents of each Revocation Data element.

When receiving a Revocation Document, the Reader SHALL process the data elements and enforce the function of the data elements. For example, for each revocation entry the Reader must update its internal revocation database.

This specification defines one data element for the Revocation Document, the Revocation Data Element defined in section 7.6.1. The Revocation Document SHALL contain one or more Revocation Data Elements.

The Revocation Data Element SHALL have the NameSpace as defined in section 7.7.

7.6.1 Revocation Data Element

The Revocation Data Element SHALL be formatted according to the following CDDL:

```
RevocationData = {
    RevocationData_Version => uint,
    RevocationData_ChangeMode => ChangeMode,
    RevocationData_Entries => [+RevocationEntry],
    RevocationData_EntriesRemove => [+RevocationEntry],
    ? RevocationData_Extensions => {+ Vendor_RegisteredID => [+ Extension] }
}

Vendor_RegisteredID = uint .size 3

ChangeMode = &(
    Overwrite : 0,
    Update    : 1
)

RevocationEntry = {
    ? RevocationEntry_PublicKeyHash => bstr,
    ? RevocationEntry_ID => bstr .size (1..16),
    ? RevocationEntry_ExpiryTime => uint .size 4
}

Extension = [
    Vendor_ExtensionID : uint,
    Version : uint,
    Data : bstr
]
```

The values for the keys in the maps SHALL be:

```
RevocationData_Version = 0
RevocationData_ChangeMode = 1
RevocationData_Entries = 2
RevocationData_EntriesRemove = 3
RevocationData_Extensions = 4

RevocationEntry_PublicKeyHash = 0
RevocationEntry_ID = 1
RevocationEntry_ExpiryTime = 2
```

7.6.2 Revocation Data Element version

The RevocationData_Version is a REQUIRED field used to indicate the version of the Revocation Data Element. The RevocationData_Version SHALL be 1. A Reader SHALL verify if the version is supported, if the value is not supported, the Revocation Data Element is considered invalid.

7.6.3 Change mode

The Revocation_ChangeMode is a REQUIRED field used to indicate the mode of the Revocation Data Element.

The RevocationData_ChangeMode SHALL be Overwrite or Update. If the RevocationData_ChangeMode is not Overwrite or Update, then the Revocation Data Element SHALL be considered invalid.

When the RevocationData_ChangeMode is set to Overwrite, the Reader SHALL overwrite the existing revocation data on the Reader for this Credential Issuer.

When the RevocationData_ChangeMode is set to Update, the Reader SHALL modify or append to the existing revocation data on the Reader for this Credential Issuer.

Where the RevocationData_ChangeMode is Overwrite, when the Revocation Document ValidityIteration is greater than the stored RevocationIteration, the Reader SHALL overwrite the stored RevocationIteration with the Revocation Document ValidityIteration.

Where the RevocationData_ChangeMode is Update, when the Revocation Document ValidityIteration is greater than the stored RevocationIteration, the Reader SHALL maintain the stored RevocationIteration.

Where the RevocationData_ChangeMode is Overwrite, when the Revocation Document ValidityIteration is equal to the stored RevocationIteration, the Reader SHALL ignore the RevocationData.

Where the RevocationData_ChangeMode is Overwrite, when the Revocation Document ValidityIteration is greater than the stored RevocationIteration, the Reader SHALL process the RevocationData.

Where the RevocationData_ChangeMode is Overwrite, all stored revocation entries with an Iteration less than the Revocation Document ValidityIteration SHALL be removed from the Reader internal revocation list for this Credential Issuer. In other words, when an Reader receives an Iteration of N with change mode of Overwrite, then delete all stored entries with an Iteration of 0 to N-1. Note: The Reader will need to store an Iteration per RevocationEntry.

Where the RevocationData_ChangeMode is Update, when the Revocation Document ValidityIteration is greater than or equal to the stored RevocationIteration, the Reader SHALL process the RevocationData.

Where the RevocationData_ChangeMode is Update, the Reader SHALL process RevocationData_Entries before processing RevocationData_EntriesRemove.

When a RevocationEntry from a RevocationData_EntriesRemove list is applied before applying RevocationData_Entries in Overwrite mode, the RevocationEntry MAY remain in the Reader internal revocation list after applying the RevocationData_Entries. The Reader is not required to maintain a separate Remove list. The condition results in access denied.

When the RevocationData_ChangeMode is Overwrite AND when the RevocationData_Entries is empty, all RevocationEntry SHALL be removed from the Reader internal revocation list.

7.6.4 Revocation entries

RevocationData_Entries and RevocationData_EntriesRemove contain the revocation entries of the Revocation Data Element. At least one of these fields SHALL be present. The RevocationData_EntriesRemove SHALL NOT be present if the change mode is set to Overwrite. The RevocationData_EntriesRemove MAY be present if the change mode is set to Update.

Each valid RevocationEntry within RevocationData_Entries SHALL be added to the Access Manager internal revocation list. Each valid RevocationEntry within RevocationData_EntriesRemove SHALL be removed from the Access Manager internal revocation list.

When updating a RevocationEntry where all possible fields are identical to an existing revocation list entry, the update SHALL be ignored. When updating a RevocationEntry where the RevocationEntry_PublicKeyHash or RevocationEntry_ID is identical to an existing revocation list entry, the existing entry shall be replaced with the received RevocationEntry.

The RevocationEntry structure SHALL contain RevocationEntry_PublicKeyHash or RevocationEntry_ID and MAY contain both. The RevocationEntry structure MAY contain the RevocationEntry_ExpiryTime field. A RevocationEntry without a RevocationEntry_ExpiryTime SHALL always be time valid. A RevocationEntry with a RevocationEntry_ExpiryTime SHALL be invalid if within a RevocationData_EntriesRemove list.

If present, the RevocationEntry_PublicKeyHash SHALL contain the SHA-256 hash of the Access Credential long term public key as per [12] (i.e. The value of the BIT STRING subjectPublicKey as uncompressed point (excluding the tag, length and number of unused bits).

If present, the RevocationEntry_ID SHALL contain the ID as defined in section 7.3.2.

If present, the RevocationEntry_ExpiryTime SHALL contain the expiration time of the revocation entry in seconds since Unix epoch. When receiving a RevocationEntry, an expired RevocationEntry_ExpiryTime SHALL invalidate the RevocationEntry, not the entiry Revocation Data Element. An example on how to use this field is to set this value to the same value as the (optional) Schedule EndTime as defined in section 7.3.4. Note that this allows the occasional cleanup of the Reader internal revocation list for any entries past the expiry time of the Access Document, limiting the memory required for the Reader internal revocation list.

The OPTIONAL RevocationData_Extensions can be used to provide extra revocation information to a Reader.

RevocationData_Extensions uses a Vendor_RegisteredID that SHALL be an IEEE OUI or CID.

7.7 DocType and NameSpace allocation

The DocType for the Access Document SHALL be "aliro-a”.

The NameSpace for the Access Data Elements SHALL be "aliro-a".

The DocType for the Revocation Document SHALL be “aliro-r”.

The NameSpace for the Revocation Data Elements SHALL be “aliro-r”.

8 Access Protocol

8.1 Transaction Overview

The Access Protocol described in this section is independent of which transport protocol is used. The different transport protocols are defined in section 9.

The transaction consists of 3 steps. The third step is OPTIONAL.

1. Transaction initiation
2. Expedited phase
3. Step-up phase

Transaction initiation are the operations required to initialize the physical and transport layer. These operations are defined as part of the transport protocol.

As described in section 5.2, a transaction then proceeds via the User Device presenting a proof of possession of a secret key, which is referred to as the **expedited phase**. If the Reader can make an access decision based on this proof, it can indicate success to the User Device and terminate the transaction. This can happen for many reasons, for example:

- The Reader has previously received information about the Access Credential via other means.
- The Reader can consult an Access Manager using the public key as a lookup.
- The Reader has previously conducted a step-up Phase and retained data for the Access Document

If the Reader cannot make an access decision based on the expedited phase, it can request the User Device for an Access Document. If available on the User Device, the Access Document contains the Access Credential public key and optionally further access information as signed data elements. The signed data elements of the Access Document can be securely transmitted between a User Device and a Reader. This is described here as the **step-up phase**.

8.1.1 Expedited Phase

The first step after transaction initiation is the expedited phase in which the User Device and Reader perform mutual or one-way authentication using the minimum number of commands and transaction time. When the Reader can make an access decision based on this authentication, the transaction can be terminated directly after the expedited phase. As a special case, it is possible for an Access Credential to be setup with key material (Kpersistent) for the expedited-fast phase directly. This can allow the User Device to perform the expedited-fast phase with a Kpersistent that is managed by a remote Access Manager.

Whether an Access Credential is explicitly selected by the user or not, it is important that no information is leaked from the User Device when a potentially malicious Reader is trying to interact with the User Device using an access transaction.

As a result, a User Device will therefore always perform the expedited phase if requested by the Reader. The authentication procedure of the expedited phase is setup in such a way, that the key material transferred between the User Device and the Reader can only be decrypted by the authentic Reader. This is intended to prevent a malicious Reader from extracting stable identifiers or to be able to detect whether the User Device has a relationship with a specific Reader.

The expedited phase can be further subdivided into two types: **expedited-standard** and **expedited-fast** phase:

8.1.1.1 Expedited-standard phase

The expedited-standard phase is intended to provide the following properties:

- Mutual authentication.
- Perfect forward secrecy.
- Tracking resilience.
- Integrity and confidentiality.

A secure channel between Reader and User Device is initiated by generating ephemeral key pairs on Reader and User Device side. A shared secret can be derived on both sides and used for generation of a shared symmetric key using Diffie-Hellman and a key derivation function.

The ephemeral public key generated on the Reader side is signed with the Reader's long term private key, which results in an authentication of the Reader by the User Device. From the User Device's perspective, the goal is that no privacy-sensitive data can be leaked by a man in the middle attack. This principle also allows the User Device to transmit data to the Reader without any leakage possibility by a passive or active eavesdropper.

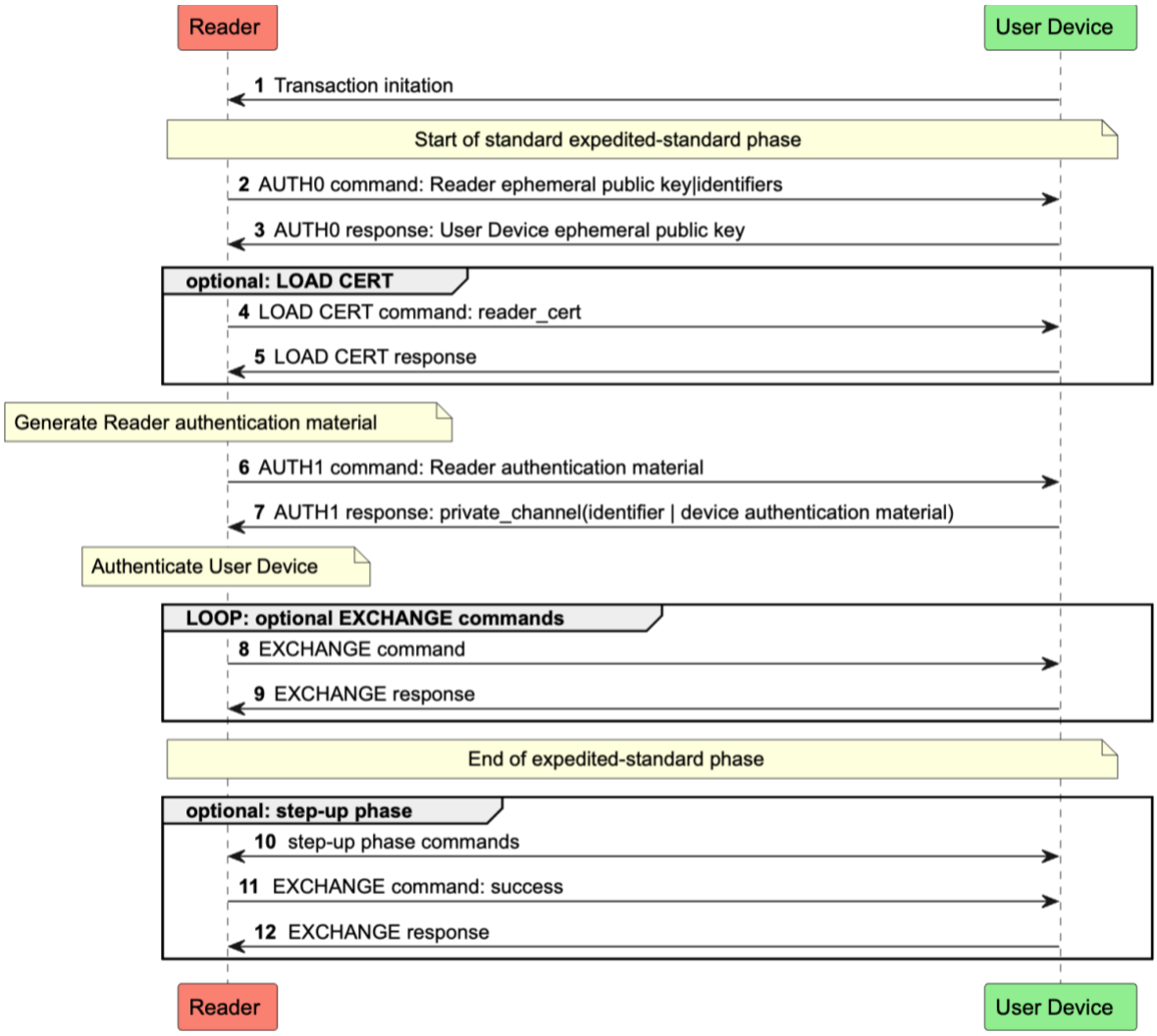
The User Device uses the established secure channel to send its public key identifier along with the signature computed on a Reader's data derived challenge and some additional application specific data. Verification of the User Device's signature by the Reader allows the Reader to authenticate the User Device.

Finally, the Reader can optionally use the EXCHANGE command to perform further operations during the expedited phase.

The User Device and Reader SHALL support the expedited-standard phase and all of the commands specified for it.

Figure 8-1 illustrates the flow of commands for the expedited-standard phase, it illustrates the scenario that results in a successful transaction.

Figure 8-1 – Expedited-standard phase Flow



8.1.1.2 Expedited-fast phase

The expedited-fast phase is intended to provide the following properties:

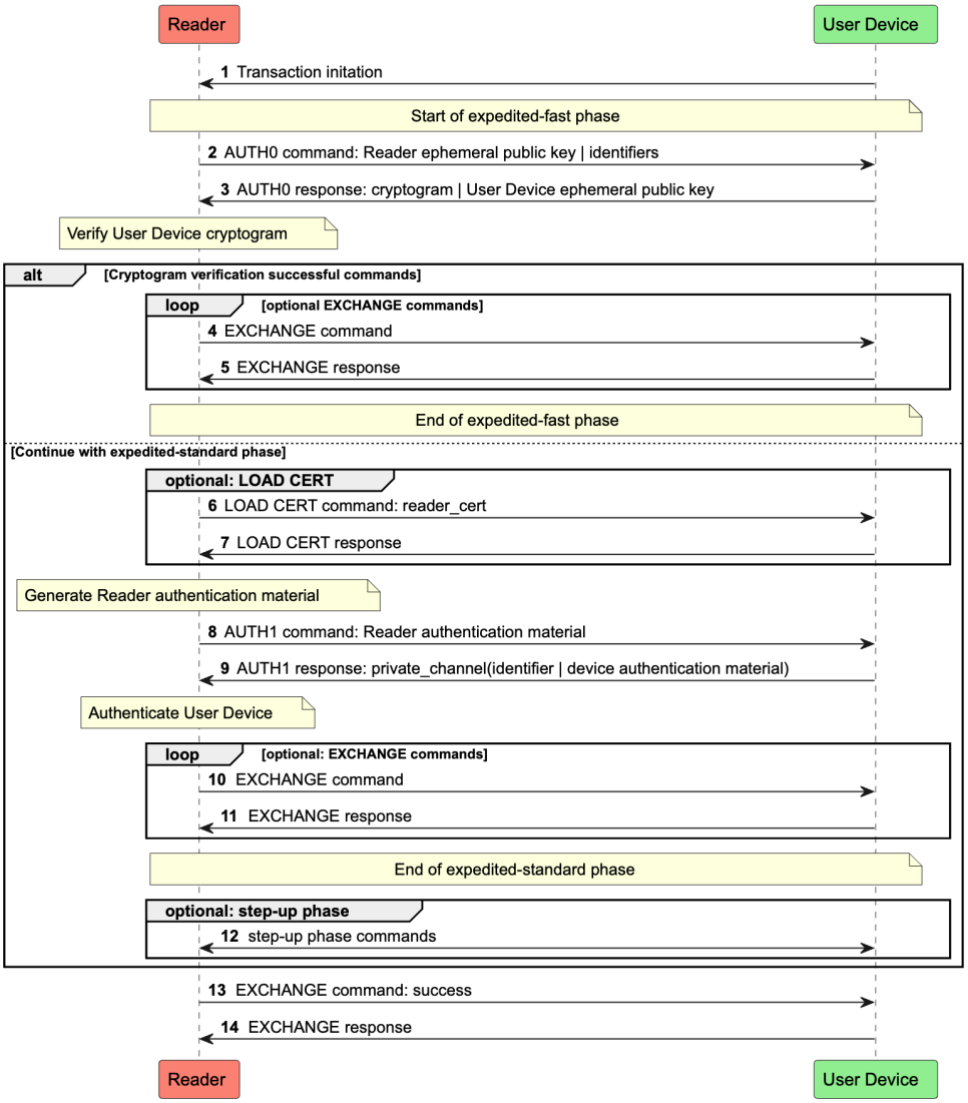
- User Device authentication or mutual authentication when EXCHANGE commands are used.
- Integrity and confidentiality.
- Tracking resilience.

The User Device generates a cryptogram based on a secret previously agreed upon during the expedited-standard phase, which allows the Reader to authenticate the User Device.

When performing the expedited-fast phase, the Reader verifies the cryptogram using trial-and-error with the different Kpersistent values it possesses. If no match is found, the Reader SHALL continue with the expedited-standard phase, except it MAY abort the transaction for security reasons.

Finally, the Reader can optionally use EXCHANGE commands to perform further operations during the expedited phase using a secure channel based on the secret used to derive the cryptogram. If this is done, the ability of the Reader to establish the secure channel authenticates the Reader to the User Device.

Figure 8-2 – Expedited-fast phase Flow



If the Reader wants to perform the step-up phase, the expedited-standard phase is required since the StepUpSK for the step-up phase is not generated as part of the expedited-fast phase.

The User Device and Reader MAY support the expedited-fast phase.

Figure 8-2 illustrates the flow of commands for the expedited-fast phase. It illustrates the scenario that results in a successful transaction.

8.1.2 Step-up phase

The step-up phase is an optional phase that can be used by the Reader and User Device to request and transmit the Access Document and/or the Revocation Document. The step-up phase can only be performed when the expedited-standard phase was used during the expedited phase.

The step-up phase uses a secure channel setup using key material derived from the expedited-standard phase. During the step-up phase the Reader can request an Access Document and Revocation Document and indicate which data elements, in these documents it requests.

Figure 8-3 – Step-up phase Flow

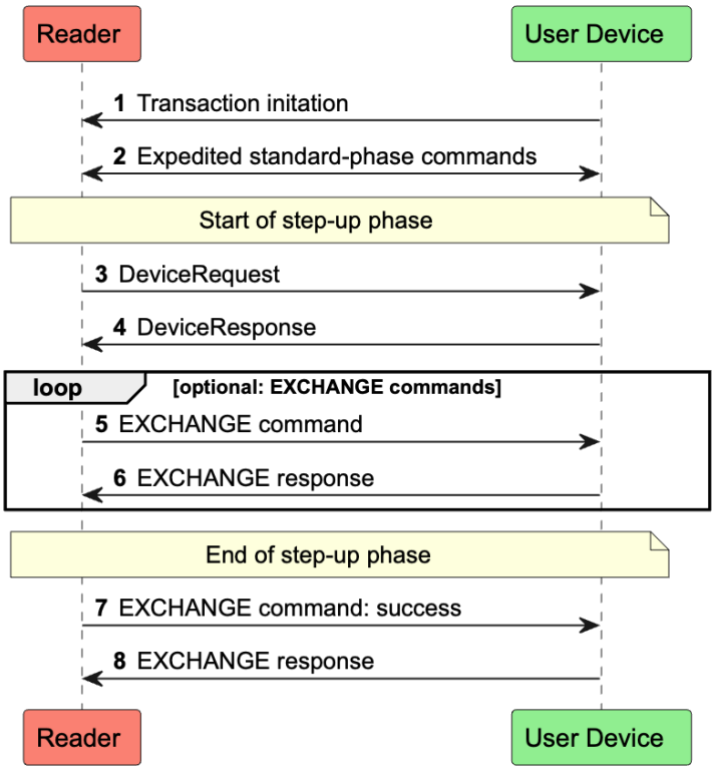


Figure 8-3 illustrates the flow of commands for the step-up phase. It illustrates the scenario that results in a successful transaction.

8.2 Access Protocol flow

This section defines the steps that the Reader and User Device SHALL follow when performing the Access Protocol. All steps SHALL be followed in order, unless otherwise indicated. Steps MAY be skipped if they are marked as OPTIONAL.

Step 1: The Access Protocol flow starts with the Reader and User Device setting up the physical and transport layer according to section 9.

Step 2: The Reader sends the AUTH0 command, see section 8.3.3.2.1. The command parameters determine if the Reader initiates the expedited-standard phase or expedited-fast phase. The User Device responds with the AUTH0 response, see section 8.3.3.2.2.

After step 2, if the Reader initiated the expedited-fast phase, there are 3 options:

- Continue with step 5.
- Continue with the expedited-standard phase by going to step 3.
- Terminate the transaction according to the transport protocol that is used.

After step 2, if the Reader initiated the expedited-standard phase it continues with step 3.

Step 3: In this OPTIONAL step, the Reader sends the LOAD CERT command, see section 8.3.3.3.1 and the User Device sends the LOAD CERT response, see section 8.3.3.3.2.

Step 4: The Reader sends the AUTH1 command, see section 8.3.3.4.1 and the User Device sends the AUTH1 response, see section 8.3.3.4.2.

Step 5: In this OPTIONAL step, the Reader and User Device exchange one or more EXCHANGE commands, see section 8.3.3.5.1, and EXCHANGE responses, see section 8.3.3.5.2.

After step 5, the expedited phase stops and the step-up phase begins. The Reader SHALL NOT continue with step 6 if it did not perform the expedited-standard phase.

Step 6: In this OPTIONAL step, the Reader and User Device exchange one or more ENVELOPE and GET RESPONSE messages and its corresponding ENVELOPE responses and GET RESPONSE responses, see section 8.4.4, as part of the step-up phase.

Step 7: In the OPTIONAL step, the Reader and User Device exchange one or more EXCHANGE command, see section 8.3.3.5.1, and EXCHANGE responses, see section 8.3.3.5.2.

Note that a transport protocol can have additional requirements on the order and optionality of certain steps.

If the User Device or Reader receives a message that does not follow the flow as required in this section it is considered a failure, and the failure process as defined in section 8.3.3.1 SHALL be executed.

8.3 Expedited phase

8.3.1 Expedited phase security

This section describes the requirements and procedures for the User Device and Reader for the cryptographic operations required for the expedited phase.

8.3.1.1 Random number generation

The User Device and Reader SHALL support random number generation. Random number generation SHOULD follow guidance from [9] or [3] for random generation and random quality evaluation.

8.3.1.2 Generate transaction data signature procedure

```
input: data_fields, private_key
output: signature
begin
  set curve_parameters ← 'ECC NIST P-256' as per [5]
  signature ← ECDSA(curve_parameters, private_key, data_fields) using SHA-256 as per [5]
  return signature (64 bytes)
end
```

8.3.1.3 Verify transaction data signature procedure

```
input: data_fields, public_key, signature
output: boolean
begin
  hash ← SHA-256(data_fields)
  set curve parameters ←'ECC NIST P-256' as per [5]
  boolean ← ECDSA_Verify(public_key, data_fields, curve_parameters, signature) using SHA-
256 as per [5]
  return boolean (true/false)
end
```

8.3.1.4 Compute shared key with Diffie-Hellman procedure

```
input: ePubK, ePrivK, transaction_identifier
output: Kdh
begin
  Compute the steps indicated by BSI TR-03111 [4] section 4.3 with the following mapping:
  Key Agreement Protocol : ECKA-DH
  (p, a, b, G, n, h) : ECC NIST P-256 Curve parameters as per [5]
  d̂ : ePrivK
  P̂: ePubK
  S_AB : key agreement output (shared secret point)
  Z_AB : computed from S_AB as per BSI TR-03111 [4] section 3.1.3
  Compute the steps indicated by the X9.63 key derivation function from BSI TR-03111 [4]
section 4.3.3 with the following inputs:
    Z_AB : Z_AB
    K : 256
    H : SHA-256
    SharedInfo : transaction_identifier
  Key derivation output Kdh: KeyData (32 Bytes)
end
```

Note that while the X9.63 key derivation function is used in the procedure to calculate Kdh, actual key derivation is performed using the procedure defined in section 8.3.1.5.

8.3.1.5 Key derivation Procedure

```
input: input_key_material, info, key_material_length, salt
output: derived_keys
begin
  Compute the steps indicated by RF 5869 [20] section 2 with the following mapping. RFC
5869 specifies a version of the extraction-expansion key derivation procedure specified in
NIST SP 800-56C [17], see note below.
  MAC algorithm : HMAC-SHA-256
  Z : input_key_material
  salt : salt
  FixedInfo : info
  L : key_material_length
```

```
  DerivedKeyingMaterial : derived_keys
end
```

Note: To perform a key derivation procedure equivalent to RFC 5869 [20], NIST SP 800-56C [17] can be used with the following configuration: the Two-Step key derivation procedure as per section 5 of [17] is selected, the MAC algorithm employed for randomness extraction is HMAC-SHA-256, the salt and shared secret Z are the same as described in the above mapping. For the key expansion procedure, the KDF in Counter Mode as described in NIST SP 800-108 [18] section 4.1 is selected and the pseudo random function is called with parameters identical to those described in RFC 5869 [20] section 2.3.

8.3.1.6 Secure channel response encryption and authentication procedure

```
input: payload, SKDevice, device_counter, aad
output: encrypted_payload, authentication_tag, device_counter
begin
  GCM-AE_K ← as defined in NIST SP 800-38D [8], using AES-256 and SKDevice as K
  IV ← 0x0000000000000001 || device_counter (unsigned big endian, 4 bytes)
  encrypted_payload, authentication_tag (16 bytes) ← GCM-AE_K(IV, payload,
    aad)
  device_counter ←  device_counter + 1
end
```

8.3.1.7 Secure channel response authentication and decryption procedure

```
input: encrypted_payload, authentication_tag, SKDevice, device_counter, aad
output: payload, authentication_tag_verified (boolean), device_counter
begin
  GCM-AD_K ← as defined in NIST SP 800-38D [8], using AES-256 and SKDevice as K
  IV ← 0x0000000000000001 || device_counter (unsigned big endian, 4 bytes)
  payload, authentication_tag_verified ← GCM-AD_K (IV, encrypted_payload,
    aad, authentication_tag)
  device_counter ←  device_counter + 1
end
```

8.3.1.8 Secure channel command encryption and authentication procedure

```
input: payload, SKReader, reader_counter, aad
output: encrypted_payload, authentication_tag, reader_counter
begin
  GCM-AE_K ← as defined in NIST SP 800-38D [8], using AES-256 and SKReader as K
  IV ← 0x0000000000000000 || reader_counter (unsigned big endian, 4 bytes)
  encrypted_payload, authentication_tag (16 bytes) ← GCM-AE_K(IV, payload,
    aad)
  reader_counter ←  reader_counter + 1
end
```

8.3.1.9 Secure channel command authentication and decryption procedure

```
input: encrypted_payload, authentication_tag, SKReader, reader_counter, aad
output: payload, authentication_tag_verified (boolean), reader_counter
```

```
begin
  GCM-AD_K ← as defined in NIST SP 800-38D [8], using AES-256 and SKReader as K
  IV ← 0x0000000000000000 || reader_counter (unsigned big endian, 4 bytes)
  payload, authentication_tag_verified ← GCM-AD_K (IV, encrypted_payload,
    aad, authentication_tag)
  reader_counter ←  reader_counter + 1
end
```

8.3.1.10 Generate AUTH0 cryptogram procedure

```
input: payload, key
output: encrypted_payload, authentication_tag
begin
  GCM-AE_K ← as defined in NIST SP 800-38D [8], using AES-256 and key as K
  IV ← 0x000000000000000000000000
  aad ← no data
  encrypted_payload, authentication_tag (16 bytes) ← GCM-AE_K(IV, payload,
    aad)
end
```

8.3.1.11 Verify AUTH0 cryptogram procedure

```
input: encrypted_payload, authentication_tag, key
output: payload
begin
  GCM-AE_K ← as defined in NIST SP 800-38D [8], using AES-256 and key as K
  IV ← 0x000000000000000000000000
  aad ← no data
  payload, authentication_tag_verified ← GCM-AD_K(IV, encrypted_payload,
    aad, authentication_tag)
end
```

8.3.1.12 Expedited-fast key material generation procedure

The User Device or Reader:

SHALL use the following values for the `flag` element: `command_parameters` || `authentication_policy` from the command data field.

SHALL use the following values for the `interface_byte`: value 0xC3 when the transaction is performed on the BLE transport or 0x5E for the NFC transport.

SHALL use the concatenation of the following values for `salt_fast`: the x component of the `reader_group_identifier_key` || the ascii string "VolatileFast" || `reader_identifier` || `interface_byte` || 0x5C || 0x02 || `current_expedited_phase_protocol_version` || the x component of the reader ephemeral public key || `transaction_identifier` || `flag` || 0xA5 proprietary information TLV according to Table 10-2 || the x component of the Access Credential public key.

SHALL use the following value for `info`: the x component of the Access Credential ephemeral public key || `auth0_command_vendor_extension` TLV (if such tag was present in AUTH0 command) || `auth0_response_vendor_extension` TLV (if such tag is present in AUTH0 response).

SHALL initialize a session-bound `expedited_device_counter` to value 0x00000001 and session-bound `expedited_reader_counter` to value 0x00000001.

SHALL generate 160 bytes of derived key material `derived_keys_fast` according to 8.3.1.5 using `Kpersistent` as `input_key_material`, `info`, and `salt_fast` as `salt`.

SHALL extract a symmetric session-bound key `CryptogramSK` from offset 0 of `derived_keys_fast` with length 32 bytes when `derived_keys_fast` is available.

SHALL extract a symmetric session-bound key `ExpeditedSKReader` from offset 32 of `derived_keys_fast` with length 32 bytes when `derived_keys_fast` is available.

SHALL extract a symmetric session-bound key `ExpeditedSKDevice` from offset 64 of `derived_keys_fast` with length 32 bytes when `derived_keys_fast` is available.

If Bluetooth LE is used as transport mechanism, SHALL extract a symmetric session-bound key `BleSK` from offset 96 of `derived_keys_fast` with length 32 bytes when `derived_keys_fast` is available.

If Bluetooth LE is used as transport mechanism and UWB ranging is supported, SHALL extract a symmetric session-bound key `URSK` from offset 128 of `derived_keys_fast` with length 32 bytes when `derived_keys_fast` is available.

Note that when the ExpeditedSK keys are available a secure channel is established.

8.3.1.13 Expedited-standard key material generation procedure

The User Device or Reader:

SHALL use the following values for the `flag` element: `command_parameters` || `authentication_policy` from the command data field.

SHALL use the following values for the `interface_byte`: value 0xC3 when the transaction is performed on the BLE transport or 0x5E when the transaction is performed on the NFC transport.

SHALL use the concatenation of the following values for `salt_volatile`: the x component of the `reader_group_identifier_key` || the ascii string "Volatile****" || `reader_identifier` || `interface_byte` || 0x5C || 0x02 || `current_expedited_phase_protocol_version` || the x component of the Reader ephemeral public key || `transaction_identifier` || `flag` || 0xA5 proprietary information TLV according to Table 10-2.

SHALL use the following value for `info`: x component of the Access Credential ephemeral public key || `auth0_command_vendor_extension` TLV (if such tag was present in AUTH0 command) || `auth0_response_vendor_extension` TLV (if such tag was present in AUTH0 response)

SHALL initialize a session-bound `expedited_device_counter` to value 0x00000001 and session-bound `expedited_reader_counter` to value 0x00000001.

SHALL generate 160 bytes of derived key material `derived_keys_volatile` according to section 8.3.1.5 using `Kdh`, `info` and `salt_volatile`.

SHALL extract a symmetric session-bound key `ExpeditedSKReader` from offset 0 of `derived_keys_volatile` with length 32 bytes.

SHALL extract a symmetric session-bound key `ExpeditedSKDevice` from offset 32 of `derived_keys_volatile` with length 32 bytes.

SHALL extract a symmetric session-bound key `StepUpSK` from offset 64 of `derived_keys_volatile` with length 32 bytes.

If Bluetooth LE is used as transport mechanism, SHALL extract a symmetric session-bound key `BleSK` from offset 96 of `derived_keys_volatile` with length 32 bytes.

If Bluetooth LE is used as transport mechanism and UWB ranging is supported, SHALL extract a symmetric session-bound key `URSK` from offset 128 of `derived_keys_volatile` with length 32 bytes.

SHALL use the concatenation of the following values for `salt_persistent`: the x component of the `reader_group_identifier_key` || "Persistent**" || `reader_identifier` || `interface_byte` || 0x5C || 0x02 || `current_expedited_phase_protocol_version` || the x component of the reader ephemeral public key || `transaction_identifier` || `flag` || 0xA5 proprietary information TLV according to Table 10-2 || the x component of the Access Credential public key.

The User Device or Reader SHOULD derive and store `Kpersistent` and the `reader_group_sub_identifier` used in the AUTH0 command for future usage in next Expedited-fast phases. If it does so, the following steps SHALL be performed by the User Device or Reader:

- Generate 32 bytes of derived key material `derived_keys_persistent` according to section 8.3.1.5 using `Kdh`, `info` and `salt_persistent`.
- Extract a symmetric key `Kpersistent` from offset 0 of `derived_keys_persistent` with length 32 bytes.
- Store `Kpersistent` and the `reader_group_sub_identifier`.

Note that when the ExpeditedSK and StepUpSK keys are available a secure channel is established.

8.3.1.14 User Authentication

User authentication can be conditionally enabled for a given Access Credential to prevent an unauthorized person from using the User Device and performing a transaction.

The means of authenticating the user are left to the User Device implementation but can include passcode or biometrics.

In case user authentication is required by the Reader or by the User Device policy but not performed by the user, the User Device SHOULD guide the user to take the steps required to perform user authentication (e.g. biometrics) and take the necessary steps to allow the transaction to be performed again. (e.g. ask user to perform an NFC tap, re-connect to the Reader of interest over Bluetooth LE…).

The User Device SHOULD offer the possibility to disable or configure the notification type related to user authentication for transactions performed over Bluetooth LE. When notifications related to user authentication are presented to the user, the User Device SHOULD inform which Access Credential or Reader is involved in the transaction being authorized by the user authentication to be performed.

During the Expedited phase the Reader SHALL indicate a user authentication policy in the AUTH0 command (see section 8.3.3.2.1). The user authentication policy indicates the Reader intent for the user authentication policy to be applied on the User Device.

The User Device will apply either a user-defined configurable policy or a hard-coded policy based on the received user authentication policy as per Table 8-1. If the Reader indicates the "Force user authentication" policy, the User Device SHALL perform User Authentication or terminate the transaction.

The Reader SHOULD NOT use the "Force user authentication" – 0x03 option when performing the Bluetooth LE + UWB Aliro Flow as defined in section 11.1.1 in a passive entry flow.

Table 8-1 – User authentication policy

Name	Value	Policy
User device setting	0x01	The Reader expects the user authentication to be requested as per user/User Device policy.
User device setting - secure action	0x02	Same as User Device setting (0x01) with the Reader indicating that it will change to a state that only further reduces the access level (i.e. "secure action"), this includes lock, arm, close etc. The User Device can use this information as extra input to decide whether to perform user authentication. The Reader SHALL NOT indicate this user authentication policy when using a Bluetooth transport protocol. This is not permitted since transmitting this field over a Bluetooth transport protocol can introduce privacy risks.
Force user authentication	0x03	The Reader requires the user authentication for this transaction without the possibility for the user/User Device to disable it.
RFU	0x00, 0x04 – 0xFF	RFU

8.3.1.15 Mailbox

The mailbox is a mechanism that allows the Reader, the Credential Issuer and/or the User Device to persistently store small data buffers. The size of mailbox is configured by the Credential Issuer during provisioning. The maximum configurable size is implementation specific.

The mailbox is accessible by the User Device and an authenticated Reader. The storage location of these persistent data is implementation specific.

The data content of the mailbox can be read/written in any order using offset/length parameters with atomicity guarantee as described in section 8.3.3.5. The offsets are relative to the start of the mailbox memory space.

Confidentiality and integrity of mailbox data exchanged between a User Device and a Reader during the transaction are in all cases protected by the currently established secure channel.

The size of the mailbox associated to an Access Credential SHALL be configurable during provisioning. The specific configuration method and size limits are outside the scope of this specification.

If the Credential Issuer uses the mailbox mechanism, the size of the mailbox SHALL be provided to the User Device by the Credential Issuer during provisioning steps. If the User Device does not support the size requested, the behavior is out of scope of this specification.

The User Device SHALL support the mailbox mechanism. The Reader MAY support the mailbox mechanism.

The mailbox SHALL NOT be used when in the step-up phase.

The mailbox data format is defined in appendix 18.

The mailbox access rights (read/write/none) SHALL be configurable and updatable by the Credential Issuer. The configuration method is out of scope of this specification. Access rights for the currently authenticated Reader are indicated during transaction in the signaling bitmap as per Table 8-11.

The mailbox content SHALL be readable and writeable by the Credential Issuer. The method used for read/write operations by the Credential Issuer is out of scope of this specification.

Read and write access to mailbox data at reader-chosen offsets is provided by the EXCHANGE command as per section 8.3.3.5. In addition, it is possible to configure an Access Credential such that the AUTH1 response returns a pre-defined subset of the mailbox. This can be useful to read a status or a table of content without an EXCHANGE roundtrip.

The offsets and lengths describing the subset of the mailbox to be returned in AUTH1 response SHOULD be provided by the Credential Issuer during provisioning steps. If no subset is provided, no data SHALL be returned in the `mailbox_data_subset`.

The mailbox subset SHALL be configurable and updatable by the Credential Issuer. The configuration method is out of scope of this specification.

8.3.2 Command format

8.3.2.1 General requirements

The command and response structures used in the expedited phase SHALL use command and response APDU structures as defined in [7].

For each command in the expedited phase, this specification specifies the instruction byte INS, the parameter bytes P1 and P2 and the command data field.

For each response in the expedited phase, this specification specifies the response data field. General requirements for the status bytes are defined in section 8.3.2.3.

The Lc and Le fields SHALL be implemented according to [7], which defines the syntax in section 5.2 of [7].

All the commands used in the expedited phase are of the proprietary class and SHALL use a class byte of 0x80 when command chaining is not used. When command chaining is used as defined in section 8.3.2.2, the requirements for the class byte apply. When command chaining is used the class byte therefore SHALL be 0x80 or 0x90.

Unless otherwise indicated, the values in the expedited-phase structures are big endian.

8.3.2.2 Oversize APDUs

The APDUs defined in this specification can result into extended and/or chained sets of APDUs depending on the size of their payloads. As a result, when chaining is used multiple APDUs can be necessary to convey a command or response payload and the procedures present in this specification MAY start after complete reception of a command or response payload.

The following rules apply:

- The User Device SHALL support receiving short length command APDUs with a command data field length up to 255 Bytes.
- The Reader SHALL support receiving short length response APDUs with a response data field length up to 256 Bytes.
- The User Device and the Reader SHALL support command chaining and response chaining for all expedited phase APDUs as defined in section 5.3 of [7] for interindustry class.
- Unless otherwise specified, the User Device SHALL be capable of receiving an APDU command where the data field length before chaining is at least 2000 Bytes.
- Unless otherwise specified, the Reader SHALL be capable of receiving an APDU response where the data field length before chaining is at least 2000 Bytes.
- The User Device and the Reader MAY support extended length APDU as defined in section 5.2 of [7]. Support and length are indicated by the User Device using a DO'7F66' element with extended length information as per section 12.8.1 of [7]. When DO'7F66' is absent, extended length is not supported by the User Device. Transmission of the DO'7F66' element is transport protocol specific, see section 10.2.1.2 and section 11.7.3.7.

8.3.2.3 General Error Conditions

This section applies to all commands of the expedited phase, it describes the generic status words to be retrieved in case of error during basic input command checking.

When 0x9000 is returned in response to a command, this is defined to mean "success".

The basic input command checking includes checking whether an INS is allowed on a given interface, checking the CLA is consistent, checking P1/P2 bytes have valid values, checking Lc is in valid range, checking the format of the payload.

The basic input command checking is executed before the steps described in listings below and can result in an error status code. It is recommended to use an error code defined in [7].

8.3.2.4 TLV Fields

The Tag Length Value fields present in the APDU commands and responses SHALL comply with the DER-TLV format as per [19] unless otherwise specified. The TLV fields SHALL be ordered as described in this specification, a different order is considered invalid unless otherwise specified.

8.3.3 Command messages

This section defines the APDU commands/responses used during the expedited phase and Step-up and the processing requirements for the User Device and Reader for those commands.

8.3.3.1 Failure process

As part of receiving and processing messages, this specification can define that a failure state has occurred. When this happens to the User Device, it SHALL:

- Return an empty response data field
- Return an error code as defined in section 8.3.2.3.
- Destroy all session-bound keys and data
- Terminate the transaction.

When this happens to the Reader, it SHALL:

- Perform Reader behavior described in Table 8-2.
- Terminate the transaction.
- Destroy all session-bound keys and data

Table 8-2 – Reader behavior when message processing results in failure state

Index	Condition 1	Condition 2	Transport	Reader behavior
1	EXCHANGE command encryption key is available	In the most recently received APDU response that is not EXCHANGE response SW = 0x9000	NFC	Send EXCHANGE command with Reader status (tag 0x97) according to section 8.3.3.5.
2			BLE	Send EXCHANGE command with Reader status (tag 0x97) according to section 8.3.3.5 and send a failure Event Message with General Error according to section 11.7.3.1
3		In the most recently received APDU response that is not EXCHANGE response SW != 0x9000	NFC	Send CONTROL FLOW command indicating failure, according to section 10.2.2
4			BLE	Send a failure Event Message with General Error according to section 11.7.3.1
5		In the most recently received EXCHANGE response SW = 0x9000 B1||B2 = 0x00	NFC	Send EXCHANGE command with Reader status (tag 0x97) according to section 8.3.3.5
6			BLE	Send EXCHANGE command with Reader status (tag 0x97) according to section 8.3.3.5 and send a failure Event Message with General Error according to section 11.7.3.1
7		In the most recently received EXCHANGE response SW = 0x9000 B1||B2 != 0x00	NFC	Send CONTROL FLOW command indicating failure, according to section 10.2.2.
8			BLE	Send a failure Event Message with General Error according to section 11.7.3.1.
9	EXCHANGE command encryption key is not available	Not applicable	NFC	Send CONTROL FLOW command indicating failure, according to section 10.2.2.
10		Not applicable	BLE	Send a failure Event Message with General Error according to section 11.7.3.1.

8.3.3.2 AUTH0 command

This command allows the Reader to initiate the authentication procedure. In case the expedited-fast phase is requested by the Reader, a cryptogram is returned allowing the Reader to proceed with verifying the cryptogram.

Note: The reader_group_sub_identifier value is a 16-byte random value picked by the Reader during installation and provided on each transaction. This value enables lookup of already established persistent symmetric key (Kpersistent) on User Device side for installations where multiple Readers share the same reader_group_identifier. The reader_group_sub_identifier does not need to be known in advance by the User Device, it is dynamically discovered at transaction time.

The reader_group_identifier is stored in the Reader during installation and is used by the User Device implementation to lookup the Access Credential and the reader_PubK to be used for the transaction.

If the validation of the cryptogram is successful and the Reader does not continue with the expedited-standard phase, the Reader SHALL make an access decision using the device key as an input.

8.3.3.2.1 Command message

The INS, P1 and P2 values of the AUTH0 command message SHALL be coded according to the following table.

Table 8-3 – AUTH0 command header

Code	Value
INS	0x80
P1	0x00
P2	0x00

The command data field of the AUTH0 command message SHALL be coded according to the following table.

Table 8-4 – AUTH0 command data field

Tag	Length (Octets)	Description	Field is
0x41	1	`command_parameters`, an unsigned big endian integer. The least significant bit has index 0. Bit0: request expedited phase: 0: Expedited-Standard phase request 1: Expedited-Fast phase request All other bits are RFU.	mandatory
0x42	1	`authentication_policy`, value as per Table 8-1	mandatory
0x5C	2	`expedited_phase_protocol_version`, the expedited protocol version selected by the Reader among the versions returned in SELECT response.	mandatory
0x87	65	`reader_ePubK` in uncompressed format starting with 0x04 followed by the x || y coordinates. The x and y coordinates are represented using 32 bytes big endian integers.	mandatory
0x4C	16	`transaction_identifier` randomly generated transaction identifier by the Reader on each transaction	mandatory
0x4D	32	`reader_identifier` field[2]. This field is composed of the concatenation of the `reader_group_identifier` (16 bytes) used for Access Credential lookup followed by `reader_group_sub_identifier` (16 bytes) used for `Kpersistent` lookup.	mandatory
0xB1	Up to 127	`auth0_command_vendor_extension` Vendor specific extension TLV as defined in Table 8-7.	optional

8.3.3.2.2 Response message

The response data field of the AUTH0 response message SHALL be coded according to the following table.

[2] When the reader_identifier field needs to be displayed in a human-readable format the recommended encoding is <reader_group_identifier>-<reader_group_sub_identifier> in hexadecimal lowercase characters.

Table 8-5 – AUTH0 response data field

Tag	Length (Octets)	Description	Field is
0x86	65	The User Device generated credential_ePubK starting with 0x04 followed by the x || y coordinates. The x and y coordinates are represented using 32 bytes big endian integers.	mandatory
0x9D	64	`cryptogram`, the authentication cryptogram (returned by the Access Credential only when the Reader selects expedited-fast phase in `command_parameters`)	conditional
0xB2	Up to 127	`auth0_response_vendor_extension` Vendor specific extensions TLV as defined in Table 8-7.	optional

8.3.3.2.3 Cryptogram payload

The cryptogram `plain_payload` SHALL be formatted according to Table 8-6.

Table 8-6 – Cryptogram payload

Tag	Length (Octets)	Description
0x5E	2	`signaling_bitmap` as defined in tag 0x5E in Table 8-11
0x91	20	`credential_signed_timestamp` as defined in tag 0x91 in Table 8-11. If no value is provided, the content of these tags SHALL be 20 bytes with value 0x00.
0x92	20	`revocation_signed_timestamp` as defined in tag 0x92 in Table 8-11. If no value is provided, the content of these tags SHALL be 20 bytes with value 0x00.

8.3.3.2.4 Vendor Specific Extensions

The vendor specific extensions TLV SHALL consist of one or more sequence TLV(s) according to [19] describing each vendor specific extension. Each of these sequence values SHALL start with an octetstring TLV according to [19] followed by other TLVs defined by the vendor. The octetstring TLV SHALL contain the three-byte IEEE OUI or CID of the vendor defining the actual vendor specific extensions.

Vendor extensions MAY not be interpreted by the receiver but SHALL always be included in the cryptography flow as per section 8.3.1.12 and section 8.3.1.13 when received.

Table 8-7 – Vendor specific extensions TLV value

Tag	Length (Octets)	Description	Field is
0x30	variable	First vendor specific extension	mandatory
0x04	3	OUI or CID for first vendor specific extension	mandatory
T1	variable	Vendor specific TLV(s)	optional
0x30	variable	Second vendor specific extension	optional
0x04	3	OUI or CID for second vendor specific extension	mandatory
T2	variable	Vendor specific TLV(s)	optional
⋮	⋮	⋮	⋮
0x30	variable	n^{th} vendor specific extension	optional
0x04	3	OUI or CID for n^{th} vendor specific extension	mandatory
T3	variable	Vendor specific TLV(s)	optional

8.3.3.2.5 Command message generation

The Reader SHALL generate the AUTH0 command message according to Table 8-3 and Table 8-4.

The Reader SHALL generate a new random ephemeral reader key pair, `reader_ePubK/reader_ePrivK`, whenever sending an AUTH0 command. This ephemeral reader key pair SHALL be ECC P-256 as specified in [5]. This key pair MAY be pre-generated before the transaction starts for performance reasons.

The Reader SHALL generate a new random transaction identifier, `transaction_identifier`.

If `expedited_phase_supported_protocol_versions` includes 0x0100, the Reader SHALL select expedited protocol version 0x0100.

8.3.3.2.6 Command message processing

The User Device SHALL generate a new random ephemeral Access Credential key pair, `credential_ePubk/credential_ePrivK`, whenever sending an AUTH0 response. This key pair SHALL be ECC P-256 as specified in [5]. This key pair MAY be pre-generated before the transaction starts for performance reasons.

The User Device SHALL execute the failure process as defined in section 8.3.3.1 if one of the following scenarios occurs:

- The AUTH0 command is not received as per sequencing flow described in section 8.2.
- The AUTH0 command is not formatted according to Table 8-3 and Table 8-4.
- The AUTH0 command contains a protocol version that is not supported.

The User Device SHALL lookup the Access Credential using the `reader_group_identifier` from the command data field.

The User Device SHOULD execute the command processing such that an external observer cannot distinguish by timing measurement whether or not the User Device possesses a given Access Credential and/or has previously established a `Kpersistent` with a Reader. See note 1 for further information on this requirement.

The User Device SHALL conditionally perform user authentication based on `authentication_policy` and following the requirements from section 8.3.1.14.

The User Device SHALL set the session-bound value `current_expedited_phase_protocol_version` using the received `expedited_phase_protocol_version`.

If the Reader requested the expedited-fast phase and the User Device supports the expedited-fast phase. The User Device SHALL lookup if it has stored an Access Credential-bound key `Kpersistent` using the `reader_group_sub_identifier`. If it finds a `Kpersistent` key:

- The User Device SHALL execute the key material generation procedure defined in section 8.3.1.12.
- The User Device SHALL compute `encrypted_payload` and `authentication_tag` as per 8.3.1.10 using `cryptogramSK` as `key` and `plain_payload` as `payload`.
- The User Device SHALL calculate the value `cryptogram` as the concatenation of `encrypted_payload` || `authentication_tag`.

If the User Device does not find a `Kpersistent` key:

- The User Device SHALL calculate the value `cryptogram` such that an external observer cannot distinguish whether or not a matching Access Credential and/or `Kpersistent` is available on the User Device by observing the returned data. See note 1 for further information on this requirement.

If the Reader requested the expedited-fast phase and the User Device does not support the expedited-fast phase:

- The User Device SHALL calculate the value `cryptogram` such that an external observer cannot distinguish whether or not a matching Access Credential and/or `Kpersistent` is available on the User Device by observing the returned data. One possible way to do this is by returning random data as the `cryptogram`.

Note 1: One possible implementation that allows the User Device that supports the expedited-fast phase to calculate the `cryptogram` such that it does not reveal possession of a given Access Credential to non-authorized Readers is by initializing a mock Access Credential with random keys (`reader_PubK`, `Kpersistent`, `credential.PubK/PrivK`).

8.3.3.2.7 Response message generation

The User Device SHALL generate the AUTH0 response according to Table 8-5.

If the Reader requested the expedited-fast phase, the User Device SHALL include the calculated `cryptogram` in the AUTH0 response.

8.3.3.2.8 Response message processing

The Reader SHALL execute the failure process as defined in section 8.3.3.1 if one of the following scenarios occurs:

- An error status word is returned by the User Device.
- The AUTH0 response is not formatted according to Table 8-5.
- A 'cryptogram' is present in the response while the AUTH0 command did not request to perform Expedited-Fast.
- A 'cryptogram' is not present in the response while the AUTH0 command did request to perform Expedited-Fast.

If the Reader requested the expedited-fast phase, it SHOULD try to decrypt the `cryptogram` received in the AUTH0 response by trying to decrypt with each stored `Kpersistent` using the following procedure:

- The Reader SHALL execute the key material generation procedure defined in section 8.3.1.12 to calculate a `cryptogramSK`.
- The Reader SHALL calculate the `plain_payload` using the procedure in section 8.3.1.11 with `cryptogramSK` as `key` and `encrypted_payload` and `authentication_tag` from the `cryptogram` in the response message as input.
- If the decryption is successful, the Reader uses the corresponding `ExpeditedSKReader`, and conditionally `BleSK` and `URSK` from the key material generation procedure.

Because verification of the `cryptogram` using the stored `Kpersistent` keys is a trial-and-error process, a limit can be imposed on this process to prevent spending excessive time with the Expedited-Fast Transaction.

Note that the trade-off on how much time to spend on this process can be calculated in advance since it is only based on the decryption time and the number of stored `Kpersistent` keys in `CryptogramSK_list`. The appropriate limits can be determined by the Reader.

The Reader can continue with the expedited-Standard phase after the Expedited-Fast phase as defined in section 8.2. This is also possible after a failed or aborted trial-and-error verification process of `authentication_tag`.

8.3.3.3 LOAD CERT command

This command allows the Reader to transfer the reader_Cert as defined in section 6.3.1. When a certificate is sent by the Reader, the Access Credential uses its associated Reader System Issuer CA certificate to verify the reader_Cert.

Alternatively, the certificate can be transferred using the remaining space of the AUTH1 command.

8.3.3.3.1 Command message

The INS, P1 and P2 values of the LOAD CERT command message SHALL be coded according to the following table.

Table 8-8 – LOAD CERT command header

Code	Value
INS	0xD1
P1	0x00
P2	0x00

The command data field of the LOAD CERT command message SHALL contain the compressed `reader_Cert` as defined in section 6.3.1.

8.3.3.3.2 Response message

The response data field of the LOAD CERT response message SHALL be empty.

8.3.3.3.3 Command message generation

The Reader MAY generate a LOAD CERT command if required by the installation configuration.

The Reader SHALL format LOAD CERT command as per section 8.3.3.3.1.

Note that the content of the LOAD CERT command is not encrypted.

The Reader SHALL generate a LOAD CERT command only if the previously executed command was AUTH0 and its response had a status word indicating success.

8.3.3.3.4 Command message processing

The User Device SHALL execute the failure process as defined in section 8.3.3.1 if one of the following scenarios occurs:

- The previous executed command was not AUTH0 or the AUTH0 response indicated failure.
- The command is not formatted as per section 8.3.3.3.1.

The User Device SHALL execute the following commands or defer the operations to AUTH1 command processing:

- Decompress and verify the reader certificate as per section 6.3.1 using the locally stored Reader System Issuer CA public key associated to the `reader_group_identifier` received in the AUTH0 command.
- Set a session-bound field `intermediate_reader_PubK` containing the subject public key present in the reader certificate if the reader certificate was successfully verified.

The User Device SHOULD execute the command processing such that an external observer cannot distinguish whether or not the User Device possesses a given reader public key or Access Credential by timing measurement.

8.3.3.3.5 Response message generation

The User Device SHALL send the LOAD CERT response message with an empty response data field.

The User Device SHALL return a LOAD CERT response such that an external observer cannot distinguish whether or not the User Device possesses a given reader public key or Access Credential by observing the returned data.

8.3.3.3.6 Response message processing

There are no LOAD CERT specific response processing requirements.

8.3.3.4 AUTH1 command

This command allows mutual authentication and establishment of a secure channel between the Reader and the User Device.

The User Device implementation SHALL allow at least 16 `reader_group_identifier` and their associated `reader_PubK` to be bound to the same Access Credential. The `reader_group_identifier` can be used to lookup the correct Access Credential. The `reader_group_sub_identifier` can be used for `Kpersistent` lookup. The `reader_group_identifier` can be used by the User Device implementation to pick the correct `reader_PubK` when more than one is associated to the selected Access Credential.

8.3.3.4.1 Command message

The INS, P1 and P2 values of the AUTH1 command message SHALL be coded according to the following table.

Table 8-9 – AUTH1 command header

Code	Value
INS	0x81
P1	0x00
P2	0x00

The command data field of the AUTH1 command message SHALL be coded according to the following table:

Table 8-10 – AUTH1 command data field

Tag	Length (Octets)	Description	Field is
0x41	1	`command_parameters`, an unsigned big endian integer. The least significant bit has index 0. Bit0: Access Credential key type request: 0: `key_slot` 1: Access Credential public key All other bits are RFU.	mandatory
0x9E	64	Reader signature	mandatory
0x90	var	`reader_Cert` as described in section 6.3.1	optional

8.3.3.4.2 Response message

The response data field of the AUTH1 response messages SHALL be the concatenation of `encrypted_payload` || `authentication_tag` as defined in section 8.3.1.6.

The unencrypted `payload` of the AUTH1 response message SHALL be formatted according to the following table:

Table 8-11 – AUTH1 response payload before encryption

Tag	Length (Octets)	Description	Field is
0x4E	8	`key_slot`, Presence depends on `command_parameters` bit 0 value.	conditional
0x5A	65	The Access Credential long term public key in uncompressed format starting with 0x04 followed by the x || y coordinates. The x and y coordinates are	conditional

Tag	Length (Octets)	Description	Field is
		represented using 32 bytes big endian integers. Presence depends on `command_parameters` bit 0 value.	
0x9E	64	User Device signature	mandatory
0x4B	variable	`mailbox_data_subset`	conditional
0x5E	2	`signaling_bitmap`, an unsigned big-endian integer. Each bit of that integer is used for signaling purpose, the least significant bit is noted Bit0. When a bit is not set, the opposite statement than what is defined below is true. Bit0: if set indicates an Access Document can be retrieved. Bit1: if set, indicates a Revocation Document can be retrieved. Bit2: if set, indicates that retrieving an Access Document or Revocation Document requires the Reader to perform the step-up AID select as defined in section 10.2. This bit is only applicable when the transaction is performed using NFC transport. When using other transport mechanisms, this bit SHALL be ignored by the Reader and SHALL not be set by the User Device. Bit3: if set, some data different from zeroes is present in the mailbox. Bit4: if set, indicates the mailbox can be read using request defined in section 8.3.3.5.1, attempts to read the mailbox SHALL return an error if not set. Bit5: if set, indicates the mailbox can be written using write/set requests defined in section 8.3.3.5.1, attempts to write or set values in the mailbox SHALL return an error if not set. Bit6: if set, sending data to the Credential Issuer backend as defined in section 8.3.3.5.1 is supported by the User Device, attempt to perform such operation when this bit is not set SHALL return an error. Bit7: if set, sending data to a bound installed application as defined in Table 8-19 is supported by the User Device, attempt to perform such operation when this bit is not set SHALL return an error.	mandatory

Tag	Length (Octets)	Description	Field is
		Bit8: Reserved for future use. Bit9: if set, indicates the User Device supports `update_doc` as defined in section 8.3.3.5.1 during the expedited-fast and expedited-standard phase. Attempts to update the access document SHALL return an error if not set. Bit10: if set indicates the mailbox feature set (read/write/set) is available in the EXCHANGE command when performed during the step-up phase. This bit SHALL be set to 0 since use of the mailbox during step-up is not supported in this version of the specification. The access rights represented by Bit4 and Bit5 apply. Bit11: if set, indicates the notify feature is supported in the EXCHANGE command when performed during the step-up phase, attempts to notify SHALL return an error if not set. Restrictions on the notify feature represented by Bit6 and Bit7 apply. Bit12: if set, indicates the User Device supports `update_doc` as defined in section 8.3.3.5.1 during the step-up phase, attempts to update the access document SHALL return an error if not set. All other bits are RFU and SHALL be ignored by the Reader.	
0x91	20	`credential_signed_timestamp` a ‘tdate’ string as defined in [6]. This field MAY be used by the User Device to convey the `signed` timestamp from the `IssuerAuth` section in the Access Document of the selected Access Credential.	optional
0x92	20	`revocation_signed_timestamp` a ‘tdate’ string as defined in [6]. This field MAY be used by the User Device to convey the `signed` timestamp from the `IssuerAuth` section in the Revocation Document of the selected Access Credential.	optional

The AUTH1 response contains either the Access Credential public key (`credential_PubK`) or a short identifier of the Access Credential public key (`key_slot`).

The `key_slot` value is shorter than the full Access Credential public key, but it requires the Reader to have storage of Access Credential public keys to look up the Access Credential public key using the `key_slot` value.

The Reader indicates in the AUTH1 command whether the User Device has to return the Access Credential public key or the `key_slot` in the AUTH1 response. The User Device SHALL return the value according to the `command_parameters` value provided in the AUTH1 command payload.

The value of the `key_slot` SHALL be the first 8 Bytes of the `keyIdentifier` computed using the Access Credential long term public key as per [12] (i.e. The `keyIdentifier` is composed of the 160-bit SHA-1 hash of the value of the BIT STRING `subjectPublicKey` as uncompressed point (excluding the tag, length, and number of unused bits)).

When the `key_slot` is used, a mechanism SHOULD be present on the Reader to deal with potential collisions on the `key_slot`.

When the mailbox subset is configured by the credential issuer, the `mailbox_data_subset` SHALL be returned in the 0x4B tag.

If no mailbox data subset is configured by the credential issuer, the 0x4B tag SHALL NOT be returned.

The Reader SHALL NOT use the `key_slot` value for any other purpose than lookup of the Access Credential public key.

8.3.3.4.3 Authentication data fields

Table 8-12 – AUTH1 Reader authentication data fields

Tag	Length (Octets)	Description	Field is
0x4D	32	`reader_identifier`	mandatory
0x86	32	`credential_ePubK.x` (unsigned big endian integer)	mandatory
0x87	32	`reader_ePubK.x` (unsigned big endian integer)	mandatory
0x4C	16	`transaction_identifier`	mandatory
0x93	4	usage = 0x415D9569	mandatory

The usage field is meant to be unique for this particular usage, providing protection against reflection attacks. It is an identifier that has no further meaning.

Table 8-13 – AUTH1 User Device authentication data fields

Tag	Length (Octets)	Description	Field is
0x4D	32	`reader_identifier`	mandatory
0x86	32	`credential_ePubK.x` (unsigned big endian integer)	mandatory
0x87	32	`reader_ePubK.x` (unsigned big endian integer)	mandatory
0x4C	16	`transaction_identifier`	mandatory
0x93	4	usage = 0x4E887B4C	mandatory

8.3.3.4.4 Command message generation

The Reader SHALL generate a AUTH1 command only if the previously executed command was AUTH0 or LOAD CERT and the response had a status word indicating success.

The Reader MAY send a reader certificate formatted as per section 13.2 if required by the installation configuration.

The Reader SHALL format the AUTH1 command as per Table 8-9 and Table 8-10.

The Reader SHALL compute Reader signature using the Reader private key over fields in Table 8-12 and according to section 8.3.1.2.

The Reader SHALL generate AUTH1 command if an access decision could not be made after processing AUTH0 response.

8.3.3.4.5 Command message processing

The User Device SHALL execute the failure process as defined in section 8.3.3.1 if one of the following scenarios occurs:

- The previously successfully executed command was neither AUTH0 nor LOAD CERT.
- The command is not formatted as per Table 8-8 and 8-9.
- A reader certificate is present in the command payload and its format is invalid.

If a reader certificate is present in the AUTH1 command, the User Device SHALL:

- Decompress and verify the reader certificate as per section 6.3.1 using a locally stored Reader System Issuer CA public key associated to the `reader_group_identifier` received in the AUTH0 command.
- Set a session-bound field `intermediate_reader_PubK` containing the subject public key present in the reader certificate if the reader certificate was successfully verified.

- Execute the failure process as defined in section 8.3.3.1 if the reader certificate signature cannot be verified with the Reader System Issuer CA public key bound to the `reader_group_identifier`.

If no reader certificate is present in the AUTH1 command and the LOAD CERT command was not sent prior to the AUTH1 command, the User Device SHALL:

- look up the reader public key using the `reader_group_identifier`.

The User Device SHALL verify the reader signature as per 8.3.1.3 computed over Table 8-12 using the `intermediate_reader_PubK` or reader public key from the previous step or LOAD CERT command processing and execute the failure process as defined in section 8.3.3.1 if the verification fails.

The User Device SHALL compute a session symmetric key `Kdh` according to section 8.3.1.4 using the previously received `transaction_identifier`, the reader ephemeral public key and the Access Credential ephemeral private key.

The User Device SHALL execute the key material generation procedure defined in section 8.3.1.13.

8.3.3.4.6 Response message generation

The User Device SHALL generate AUTH1 response according to section 8.3.3.4.2.

The User Device SHALL return an AUTH1 response such that an external observer cannot distinguish whether or not the User Device possesses a given reader public key or Access Credential by observing the returned data.

An example of the implementation of this requirement is that the User Device has to return the same error code whether the Reader signature or Reader certificate could not be verified because no public key exists on the User Device for verification or because the public key exists, but the Reader signature or Reader certificate verification failed.

The User Device SHOULD execute the command processing such that an external observer cannot distinguish whether or not the User Device possesses a given reader public key or Access Credential by timing measurement.

The User Device SHALL compute User Device signature using the Access Credential private key over fields in Table 8-13 and according to section 8.3.1.2.

The User Device SHALL generate `encrypted_payload` and `authentication_tag` according to section 8.3.1.6 using the session-bound key `ExpeditedSKDevice` as `SKDevice`, the data to be returned formatted as per Table 8-11 as payload, `expedited_device_counter` as device_counter, an empty field as aad. The device_counter output from the procedure is used to update `expedited_device_counter`.

The User Device SHALL return the concatenation of `encrypted_payload` ‖ `authentication_tag` as response payload if no error was triggered.

8.3.3.4.7 Response message processing

The Reader SHALL execute the failure process as defined in section 8.3.3.1 if one of the following scenarios occurs:

- An error status word is returned by the User Device.
- The AUTH1 response is not formatted according to 8.3.3.4.2.

The Reader SHALL execute the key material generation procedure defined in section 8.3.1.13.

The Reader SHALL compute a session symmetric key `Kdh` according to section 8.3.1.4 using the Access Credential ephemeral public key as `ePubK`, the reader ephemeral private key as `ePrivK`, `transaction_identifier`.

The Reader SHALL verify `authentication_tag` and decrypt `encrypted_payload` according to section 8.3.1.7 using `ExpeditedSKDevice` as `SKDevice`, `expedited_device_counter` as device_counter, an empty field as aad. The device_counter output from the procedure is used to update `expedited_device_counter`.

The Reader SHALL execute the failure process as defined in section 8.3.3.1 if `authentication_tag` cannot be verified.

The Reader SHALL try to lookup the Access Credential public key using the `key_slot` provided in the response.

The Reader SHALL verify the User Device signature computed over Table 8-13 if provided in the response or defer this verification to step-up phase. The Reader SHALL NOT use any information for the access decision before verification of the User Device signature.

If the validation of the User Device is successful and the Reader does not continue with the step-up phase, the Readers' access decision SHALL use the Access Credential public key as input.

8.3.3.5 EXCHANGE command

This command allows the Reader to perform various operations on the Access Credential during the transaction expedited-fast, expedited-standard and step-up phase. See the signaling bitmap in section 8.3.3.4.2 for details on the supported items during step-up.

The command and response payloads are always wrapped in the currently established secure channel.

This command reads, writes, or sets data from a mailbox.

The command also allows to send transient notification messages to the User Device.

The 'set' request fills an arbitrary mailbox area with the provided single byte value.

Multiple 'read', 'write', 'set' operations can be present in a single EXCHANGE command. This is done by including multiple mailbox commands in a single 0xBA tag.

Multiple 'notify' operations can be present in a single EXCHANGE command. This is done by including multiple notify commands in a single 0xAE tag.

All mailbox write/set operations contained in an EXCHANGE command are written atomically and in the order they appear in the input buffer by default when no ‘Open Atomic session’ indicator is set.

All mailbox read operations contained in an EXCHANGE command are returned in the order they appear in the request buffer.

When multiple EXCHANGE commands are part of an atomic session, all written mailbox values are effectively readable only after successful execution of a command with ‘End Atomic Session’ indicator set, all mailbox read operations occurring before that point return the old value.

An atomic session spanned over multiple commands is executed by setting the ‘Atomic Session’ indicator flag to 1 on a series of consecutive EXCHANGE commands, the session is closed and all mailbox data written by setting the Atomic Session indicator flag to 0 on the last EXCHANGE command of the series. If an atomic session is not closed using the Atomic Session indicator, mailbox data that is part of the open atomic session SHALL NOT be written.

When there are multiple write/set operations contained in an atomic session spanning multiple EXCHANGE commands, they are performed in the order in which they are received within a single EXCHANGE command first, then in the order in which the EXCHANGE commands are received within the atomic session.

The atomic session requirements only apply to the mailbox commands in Table 8-16. Therefore if an atomic session has started and a subsequent exchange command does not contain tag 0xBA the atomic session will be kept open.

When an atomic session is open and a command is received that is not an EXCHANGE command, the User Device SHALL execute the failure process as defined in section 8.3.3.1.

The User Device SHALL support writing all the bytes of the mailbox within an atomic session potentially spanned over multiple EXCHANGE commands.

When using NFC for the transaction protocol, the EXCHANGE command is used by the reader to indicate the end of the transaction and the final Reader state using tag 0x97. When using BLE for transaction protocol, tag 0x97 is present in the EXCHANGE command sent by Reader only if the transaction failure occurs (see Table 8-15). Otherwise, Tag 0x97 is absent in the EXCHANGE command when using BLE for transaction protocol. Tag 0x97 MAY be combined with other tags in a single EXCHANGE command. After an EXCHANGE command containing 0x97, no further EXCHANGE commands can be sent.

The EXCHANGE command uses the ExpeditedSKDevice and ExpeditedSKReader when in the expedited phase and StepUpSKDevice and StepUpSKReader when in the step-up phase.

8.3.3.5.1 Command message

The INS, P1 and P2 values of the EXCHANGE command message SHALL be coded according to the following table.

Table 8-14 – EXCHANGE command header

Code	Value
INS	0xC9
P1	0x00
P2	0x00

The command data field of the EXCHANGE command message SHALL be the concatenation of `encrypted_payload || authentication_tag` as defined in section 8.3.1.8.

The unencrypted `payload` of the EXCHANGE response message SHALL be formatted according to the following table:

Table 8-15 – EXCHANGE command payload before encryption

Tag	Length (Octets)	Description	Field is
0xBA	variable	Mailbox commands, see Table 8-16. Only a single 0xBA command can be present. Multiple read/write/set mailbox commands MAY be nested under 0xBA.	optional
0xAE	variable, max 250 Bytes	notify User Device with data present in this field. The content of this field is one or more of the following DER-TLV tags. Only a single 0xAE command can be present Tag = 0x82, Len=2, value=reader error code as 16-bits big endian unsigned integer, see Table 8-18. Tag = 0xB5, with variable length, reader descriptor see Table 8-17 Tag = 0x9FXX with variable length, notify Credential Issuer backend or application, see Table 8-19. Multiple tags MAY be nested under 0xAE. Only one instance of 0x82, 0xB5, and 0x9FXX SHALL be nested in a single 0xAE.	optional
0x97	2	Reader status as per Table 8-18. This tag SHALL be accepted on the NFC interface by the User Device. This tag SHALL be accepted by the User Device on the BLE interface only if the first byte is	Conditional

Tag	Length (Octets)	Description	Field is
		set to 0x00. Otherwise, this tag SHALL NOT be sent and received on the BLE interface. Upon reception of this tag, the User Device MAY inform the user the transaction has concluded and MAY close the NFC or BLE transport after sending back the EXCHANGE response.	
0x98	0	Make URSK available upon request. This tag is accepted only on the Bluetooth LE interface once per expedited phase.	optional
0x81	variable	`update_doc`, Request the User Device to update/replace an existing Access Document using the data contained in this tag, data format is part of provisioning and out of scope of this specification. Support for that feature by the User Device is indicated in the `signaling_bitmap`. The updated Access Document SHALL be retrievable on the next transaction. The User Device SHALL return an error if this tag is sent more than once per transaction.	optional

Table 8-16 – Mailbox commands

Tag	Length (Octets)	Description	Field is
0x8C	1	single option byte: bit0 Atomic Session indicator start(1)/stop(0), other bits RFU. Only a single 0x8C tag can be present	mandatory
0x87	4	offsetmsb || offsetlsb || lengthmsb || lengthlsb, read request in mailbox	Conditional, present if read request.
0x8A	variable	offsetmsb || offsetlsb || data, write request in mailbox	Conditional, present if write request.
0x95	5	offsetmsb || offsetlsb || lengthmsb || lengthlsb || value, set request in mailbox	Conditional, present if set request.

Table 8-17 – Reader Descriptor

Tag	Length (Octets)	Description	Field is
0xB5	variable	Reader Descriptor	Optional
.. 0x04	3	Reader Vendor ID	Mandatory

Tag	Length (Octets)	Description	Field is
.. 0x80	Variable	Reader Product ID	Mandatory
.. 0x81	variable	Reader Firmware Version	Mandatory

Support for the reader information field is optional for the Reader to implement and to use in a configuration.

Reader Vendor ID identifies a Reader product manufacturer. This field is 3 octets in length and its value is the IEEE OUI or CID of the Reader vendor. Reader Product ID is variable length field that identifies a product of a Reader vendor. The Reader Product ID is assigned by the Reader vendor. Reader Firmware Version is variable length field that identifies the software firmware version executing on the Reader.

Table 8-18 – Reader status

First byte	Second byte	Meaning
0x00	0x01	Access Credential public key not found
	0x02	Access Credential public key expired
	0x03	Access Credential public key not trusted
	0x04	Invalid User Device signature
	0x06	invalid data format
	0x07	invalid data content
	0x20	status word error
	0x21	no key slot in response
	0x22	no public key in response
	0x23	no User Device signature present
	0x25	invalid access rights
	0x26	hardware issue
	Other values	reserved for future use
0x01	0x00	indicates that the Reader state is secure. Secure includes locked, armed, closed etc.
	0x01	indicates that the Reader state is unsecure. Unsecure includes unlocked, disarmed, opened etc.
	0x02	indicates that the Reader state is obstructed/jammed/stuck. For e.g., deadbolt is partially extended and not latched properly.
	0x80	indicates the Reader has started the operation to enter the secure state
	0x81	indicates the Reader has started the operation to enter the unsecure state
	0x82	Unknown – Reader state is not available at the Reader
	Other values	reserved for future use

Tag 0xAE with sub tag 0x9Fxx can contain data for the Credential Issuer backend or application. The xx in sub tag 0x9Fxx SHALL be coded according to the following table:

Table 8-19 – 0x9Fxx tag encoding

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Meaning
0	0	1	-	-	-	-	-	request to send data present in this tag to a bound application[3] installed on the User Device.
0	1	0	-	-	-	-	-	request to send data contained in this tag to the Credential Issuer backend[4].
0	1	0	1	-	-	-	-	request is time sensitive, the User Device should try to deliver immediately to the Credential Issuer backend
0	1	0	0	-	-	-	-	request is not time sensitive and may be sent in batch later to the Credential Issuer backend
0	1	0	-	-	x	x	x	Importance level from 0 to 4.

The notification mechanism is best effort in nature. To help the User Device save battery and network usage when resources are constrained, the Reader can indicate the time sensitiveness and the subsampling rate to be applied for a notification. When subsampling value is set to less than 100% the Reader hints at the User Device that some portion of the notifications received may be dropped by the User Device. When its resources are limited, that value could be set by the Reader based for example on the expected number of transactions per day at a given Reader location.

Table 8-19 specifies Importance levels from 0 to 4 and the associated subsampling rate:

- Critical (4), the User Device delivers 100% of these messages even in special situation like low battery.
- High (3), the User Device delivers at least 75% of these messages.

[3] The behavior upon reception of the data by the bound application and the method used by the installed application to retrieve the data is dependent on the user device platform and out of scope of this specification. The binding between a User Device Access Credential and the application receiving the data is part of the provisioning and out of scope of this specification.

[4] configuration of this functionality is part of provisioning and out of scope of this specification.

- Medium (2), the User Device delivers at least 50% of these messages.
- Low (1), the User Device delivers at least 25% of these messages.
- Very Low (0), the User Device delivers at least 5% of these messages.

8.3.3.5.2 Response message

The response data field of the EXCHANGE response messages SHALL be the concatenation of `encrypted_payload || authentication_tag` as defined in section 8.3.1.6.

The unencrypted `payload` of the EXCHANGE response message SHALL be formatted according to the following table. The length can be from 0 to 65535, it is encoded on 2 bytes using the big endian convention (lengthmsb || lengthlsb).

Table 8-20 – EXCHANGE response payload before encryption

Length (Octets)	Description	Field is
variable	data read from request 1	conditional
variable	data read from request 2	conditional
variable	… data read from request n	conditional

8.3.3.5.3 Command message generation

The Reader SHALL generate the EXCHANGE command according to section 8.3.3.5.1.

The Reader SHALL generate an EXCHANGE command only if the previously successfully executed command was AUTH0, AUTH1, EXCHANGE or ENVELOPE (step-up).

When in the expedited phase, the Reader SHALL generate `encrypted_payload` and `authentication_tag` according to section 8.3.1.8 using `ExpeditedSKReader` as `SKReader`, the command payload formatted as per Table 8-15 as payload, `expedited_reader_counter` as `reader_counter`, an empty field as aad. The reader_counter output from the procedure is used to update `expedited_reader_counter`.

When in the step-up phase, the Reader SHALL generate `encrypted_payload` and `authentication_tag` according to section 8.3.1.8 using `StepUpSKReader` as `SKReader`, the command payload formatted as per Table 8-15 as payload, `StepUp_reader_counter` as `reader_counter`, an empty field as aad. The reader_counter output from the procedure is used to update `StepUp_reader_counter`.

8.3.3.5.4 Command message processing

The User Device SHALL execute the failure process as defined in section 8.3.3.1 if one of the following scenarios occurs:

- The EXCHANGE command is not received as per sequencing flow described in section 8.2.

- The EXCHANGE command is not formatted according to section 8.3.3.5.1.
- The session-bound `expedited_device_counter` or `StepUp_device_counter` value or the session-bound `expedited_reader_counter` or `StepUp_reader_counter` value is 0xFFFF before any increment occurring in the command processing.

When in the expedited phase, the User Device SHALL verify `authentication_tag` and decrypt `encrypted_payload` according to section 8.3.1.9 using `ExpeditedSKReader` as `SKReader`, `expedited_reader_counter` as reader_counter, an empty field as aad. The reader_counter output from the procedure is used to update `expedited_reader_counter`.

When in the step-up phase, the User Device SHALL verify `authentication_tag` and decrypt `encrypted_payload` according to section 8.3.1.9 using `StepUpSKReader` as `SKReader`, `StepUp_reader_counter` as reader_counter, an empty field as aad. The reader_counter output from the procedure is used to update `StepUp_reader_counter`.

The User Device SHALL execute the failure process as defined in section 8.3.3.1 if `authentication_tag` cannot be verified.

The User Device SHALL execute all the requests as defined in Table 8-20.

The User Device SHALL atomically execute all the write and set requests contained in a command even if no atomic session has been started.

8.3.3.5.5 Response message generation

The User Device SHALL generate EXCHANGE response according to section 8.3.3.5.2.

The User Device SHALL return in the response payload as part of the encrypted channel the byte array 0x0002||B1||B2 and no other data during processing of a read/write or set request. An example of such an error is an out-of-bound request. B1, B2 are implementation specific error codes.

The User Device SHALL return in the response payload as part of the encrypted channel the byte array 0x0002||B1||B2 and no other data if an error occurred during processing of the notify requests. B1, B2 are implementation specific error codes.

The User Device SHALL return in the response payload as part of the encrypted channel the byte array 0x0002||B1||B2 and no other data if an error occurred during processing of the `update_doc` or `update_doc` feature is not available on the User Device or no access document already exists for that Access Credential. B1, B2 are implementation specific error codes.

The User Device SHALL return in the response payload as part of the encrypted channel the byte array 0x0002||B1||B2 and no other data if an error occurs during processing of the 0x98 tag. Examples of such errors are when tag 0x98 is present more than once in the current command or current transaction. B1, B2 are implementation specific error codes.

The User Device SHALL append at the end of the response payload as part of the encrypted channel the byte array 0x0002||0x00||0x00 when all requests present in EXCHANGE command executed successfully.

When in the expedited phase, the User Device SHALL generate `encrypted_payload` and `authentication_tag` according to section 8.3.1.6 using the session-bound key `ExpeditedSKDevice` as `SKDevice`, the data to be returned formatted as per Table 8-20 as payload, `expedited_device_counter` as `device_counter`, and empty field as aad. The device_counter output from the procedure is used to update `expedited_device_counter`.

When in the step-up phase, the User Device SHALL generate `encrypted_payload` and `authentication_tag` according to section 8.3.1.6 using the session-bound key `StepUpSKDevice` as `SKDevice`, the data to be returned formatted as per Table 8-20 as payload, `StepUp_device_counter` as device_counter, and empty field as aad. The device_counter output from the procedure is used to update `StepUp_device_counter`.

The User Device SHALL destroy all session-bound keys and data after a sequence 0x0002||B1||B2 indicating an error is returned in the secure channel.

8.3.3.5.6 Response message processing

The Reader SHALL execute the failure process as defined in section 8.3.3.1 if one of the following scenarios occurs:

- An error status word is returned by the User Device.
- The EXCHANGE response is not formatted according to section 8.3.3.5.2.

When in the expedited phase, the Reader SHALL verify `authentication_tag` and decrypt `encrypted_payload` according to section 8.3.1.7 using `ExpeditedSKDevice` as `SKDevice`, `expedited_device_counter` as device_counter, an empty field as aad. The device_counter output from the procedure is used to update `expedited_device_counter`.

When in the step-up phase, the Reader SHALL verify `authentication_tag` and decrypt `encrypted_payload` according to section 8.3.1.7 using `StepUpSKDevice` as `SKDevice`, `StepUP_device_counter` as device_counter, an empty field as aad. The device_counter output from the procedure is used to update `StepUp_device_counter`.

The Reader SHALL execute the failure process as defined in section 8.3.3.1 if `authentication_tag` cannot be verified.

8.4 Step-up phase

8.4.1 Overview

The purpose of the step-up phase is to transfer the Access Document and/or the Revocation Document between the User Device and the Reader. The step-up phase is based on [6], when that specification is referenced, the User Device is the mdoc and the Reader the mdoc reader.

The Reader MAY support the step-up phase, the User Device SHALL support the step-up phase.

The Access Document and Revocation Document can be requested and transmitted using the DeviceRequest and DeviceResponse messages as specified in [6]. The DeviceRequest message will contain the Access Data Element identifiers that the Reader wants to request. The DeviceResponse message will contain the returned documents containing the data elements that the User Device can return.

These messages are encrypted using the Session Encryption mechanism and encapsulated using SessionData messages.

The SessionData messages are then transmitted using the ENVELOPE and GET RESPONSE APDUs also defined in [6].

Note that section 7.2 requires all CBOR structures to be encoded according to the core deterministic requirements in section 4.2.1 of [27].

After sending the SessionData messages the Reader and User Device MAY exchange one or more EXCHANGE commands, see section 8.3.3.5.1, and EXCHANGE responses, see section 8.3.3.5.2.

8.4.2 Request and Response messages

If the Reader wants to request an Access Document and/or a Revocation Document, it SHALL use DeviceRequest messages as defined in [6]. The document type, namespaces and data element identifiers of the requested documents are defined in section 7.

The User Device SHALL use DeviceResponse messages defined in [6] to return the Access Document and/or the Revocation Document.

User Devices SHOULD implement logic to only return data elements that are requested. To determine which data elements to return, User Devices are expected to use exact string match of the DataElementIdentifier. User Devices MAY use other logic to determine which data elements to return.

Note: An Access Document can contain multiple of the data elements that are requested. The content of the data element and other logic used by the Reader may prevent a data element from granting access when verified by the Reader. Therefore, it is RECOMMENDED to return all data elements that are requested and present.

The User Device SHOULD NOT return any data elements in the Access Document if the Validity structure in IssuerAuth is not valid.

If the User Device does not return any data elements, it SHALL return a DeviceResponse message without the document field.

The reader authentication and mdoc authentication mechanisms defined in [6] are not used. The following additional requirements apply to the presence of fields in the DeviceRequest and DeviceResponse messages:

- In the DeviceRequest: readerAuth SHALL NOT be present and requestInfo SHOULD NOT be present.
- In the DeviceResponse: documentErrors, errors and deviceSigned SHALL NOT be present.
- To optimize the size of the DeviceRequest structure the keys in the maps SHALL be replaced by a different key (note that these are integers, still encoded as text strings) according to Table 8-21. Some of the keys are nested, shown by indentations.
- To optimize the size of the DeviceResponse structure the keys in the maps SHALL be replaced by a different key (note that these are integers, still encoded as text strings) according to Table 8-22. Some of the keys are nested, shown by indentations.

Table 8-21 – New DeviceRequest key values

Original key	New key
“version”	“1”
“docRequests”	“2”
“itemsRequest”	“1”
“nameSpaces”	“1”
“requestInfo”	“2”
“docType”	“5”

Table 8-22 – New DeviceResponse key values

Original key	New key
“version”	“1”
“documents”	“2”
“issuerSigned”	“1”
“nameSpaces”	“1”
“IssuerAuth”	“2”
“docType”	“5”
“status”	“3”

8.4.3 Session encryption

The DeviceRequest and DeviceResponse messages SHALL be encapsulated in the SessionData message structure as defined in [6] clause 9.1.1.4. The SessionData messages SHALL NOT contain the “status” field.

The cryptographic operations for Session Encryption as defined in [6] clause 9.1.1.5 SHALL be implemented with the following changes:

- The cryptographic operations in [6] clause 9.1.1.5 SHALL use the 32 bytes value for StepUpSK computed as per section 8.3.1.13 as the IKM.
- The cryptographic operations in [6] clause 9.1.1.5 SHALL use an empty salt.

The result of these cryptographic operations is to create `StepUpSKDevice` and `StepUpSKReader` to use as `SKDevice` and `SKReader` and to create `StepUp_device_counter` and `StepUp_reader_counter` to use as `device_counter` and `reader_counter`.

If an error occurs, the procedure in section 8.3.3.1 SHALL be executed.

8.4.4 APDU commands

The SessionData messages SHALL be transmitted using the ENVELOPE and GET RESPONSE commands defined below. In this specification, the User Device indicates the supported APDU command and response sizes using the SELECT response as defined in section 10.2.1.2.

8.4.4.1 ENVELOPE

The ENVELOPE command SHALL be coded according to structure defined in [6].

8.4.4.2 GET RESPONSE

The GET RESPONSE command SHALL be encoded as per structure defined in section 11.7.1 of [7].

8.4.4.3 EXCHANGE

The EXCHANGE command and response are defined in section 8.3.3.5.

9 Transport Protocols

The Transport Protocols define the requirements to setup the connections, transmit the Access Protocol messages and perform Transport Protocol specific functions.

A transaction can take place using either NFC or Bluetooth LE as the transport mechanism for the access protocol. When performing the transaction over Bluetooth LE, UWB can be used to securely determine proximity.

This specification defines 3 different flows:

1. Using NFC as the transport protocol.
2. Using Bluetooth LE as the transport protocol and UWB for securely determining proximity between the User Device and the Reader.
3. Using Bluetooth LE as the transport protocol and explicit selection of the Reader and access action by the user.

For 1: the User Device and the Reader SHALL support this flow according to the requirement in section 10.1.

For 2: the User Device and the Reader MAY support this flow according to the requirements in section 11.1.1.

For 3: the User Device and the Reader MAY support this flow according to the requirements in section 11.1.2.

10 NFC

10.1 Reader and User Device requirements

When NFC is used as a transport layer, the following requirements apply:

- The Reader SHALL be able to operate in Poll Mode.
- The Reader SHALL be able to poll for Technology Type NFC-A and SHALL support T4AT Platform and ISO-DEP protocol as defined in [22].
- The User Device SHALL be able to operate in Listen mode.
- The User Device SHALL be able to respond in Technology Type NFC-A and SHALL support T4AT Platform and ISO-DEP protocol as defined in [22].

The Reader SHOULD NOT show a success indication if an EXCHANGE command indicating success has not yet been sent.

The Reader SHOULD NOT show a failure indication if the transaction has not been terminated and a CONTROL FLOW command has not yet been sent.

The Reader can proceed with triggering the mechanical actuation as soon as an access decision has been made and independently of the result of subsequent APDU exchanges like EXCHANGE or CONTROL FLOW.

10.2 Transaction

When the transaction is performed using NFC, the Access Protocol commands SHALL be transmitted using NFC following the flow as defined in section 8 with the following additional requirements:

Before sending the AUTH0 command as part of the Access Protocol, the Reader SHALL send the SELECT command as defined in section 10.2.1 with the expedited phase AID.

The User Device can indicate that it is required to send a step-up AID SELECT command before the step-up phase. This can be useful for example if selection of a different AID is required to get access to the Access or Revocation Document. The need for a SELECT before the Step-up phase is defined by the `signaling_bitmap` returned during the expedited phase in the AUTH1 response payload. If the User Device uses this mechanism to indicate the SELECT is required, the Reader SHALL send the step-up AID SELECT command before performing the step-up phase. Note that this is between step 5 and step 6 as defined in section 8.2.

When transitioning from Expedited phase to step-up phase and whether or not a step-up AID is performed, the same NFC transport session is used for both phases. Therefore, either side SHALL NOT perform intentional protocol deactivation.

When supported and configured for use, Reader Descriptor information SHALL be sent by the Reader once during an NFC Access Protocol transaction, in the first EXCHANGE or CONTROL FLOW command.

After completing the Access Protocol as defined in section 8, the Reader SHALL send the EXCHANGE command with a Reader status sub-event defined in section 8.3.3.5, or the CONTROL FLOW command as defined in section 10.2.2 when the secure channel is not available, to indicate the result of the transaction. After the response to one of these commands is received:

- The User Device and Reader SHALL destroy all the session data of the current transaction.
- The transaction can be terminated

Note that [22] describes the process for termination (deactivation) of the NFC protocol.

The commands defined in this section SHALL implement the requirements from section 8.3.2, with the exception of the class byte requirements for the SELECT command, which is defined in section 10.2.1.1.

The Reader SHOULD NOT show a success indication if EXCHANGE command indicating success has not been sent.

10.2.1 SELECT

The SELECT command is used by the Reader to select an application instance using the AID.

10.2.1.1 Command Message

The CLA, INS, P1 and P2 values of the SELECT command message SHALL be coded according to the following table.

Table 10-1 – SELECT command header

Code	Value
CLA	0x00
INS	0xA4
P1	0x04
P2	0x00

The command data field of the SELECT command message SHALL contain the AID of the application to be selected as defined in Table 10-3.

10.2.1.2 Response Message

The response data field of the SELECT response message SHALL be coded according to the following table.

Table 10-2 – SELECT response message

Tag	Length (Octets)	Description	Field is
0x6F	variable	File Control Information (FCI)	mandatory
..0x84	9	AID	mandatory

Tag	Length (Octets)	Description	Field is
..0xA5	variable	Proprietary Information	mandatory
….0x80	2	Type	mandatory
….0x5C	2 x n	expedited_phase_supported_protocol_versions	conditional (expedited phase only)
….0x7F66	8	Extended length information	conditional
……0x02	2	Maximum command APDU	mandatory
……0x02	2	Maximum response APDU	mandatory
….0xB3	variable, up to 127	Vendor specific extensions	optional
0xB7	variable	User Device Descriptor	optional
.. 0x04	3	User Device Vendor ID	mandatory
.. 0x80	variable	User Device Product ID	mandatory
.. 0x81	variable	User Device Firmware Version	mandatory

To allow forward compatibility and future protocol upgrades the Reader SHALL accept any list of protocols (see expedited_phase_supported_protocol_versions below) even if some of the listed protocol versions are not known to the Reader as long as at least one of these protocols is supported by the Reader.

The size of the SELECT response message SHALL NOT exceed 256 bytes.

The FCI template follows the definition of [7]. For forward compatibility, the Reader SHALL accept unknown tag values present in the FCI template tree, when such tags are present in the FCI the Reader SHALL NOT interpret their content.

The AID TLV SHALL contain the AID of the selected application. The AIDs are defined in Table 10-3:

Table 10-3 – AIDs

Application	AID
Expedited Phase	A000000909ACCE5501
Step-up Phase	A000000909ACCE5502

The Type TLV SHALL contain the application type Aliro application according to Table 10-4. The value of the application type is coded in big endian format.

Table 10-4 – Application types

Type value	Application type
0x0000	CSA application
0x0001 – 0xFFFF	RFU

If the selected application supports extended length APDUs, the proprietary information TLV in the FCI template SHALL also include the extended length information TLV according to [7]. This TLV shall include two integer TLVs with big endian notation according to [19]. The first integer TLV SHALL contain the maximum number of bytes in a command APDU. The second integer TLV SHALL contain the maximum N_e for the response APDU according to [7].

If the Expedited Phase AID was selected, the proprietary information TLV in the FCI template SHALL also include the expedited_phase_supported_protocol_versions TLV. This TLV SHALL contain the list of supported protocol versions ordered from highest to lowest, each version number is concatenated and encoded on 2 bytes in big endian notation. (at least 0x0100 is present in the list, this value represents the expedited phase protocol version defined in this specification).

A vendor specific extensions TLV as described in section 8.3.3.2.3 MAY be added to the proprietary information TLV in the FCI template.

Support for the User Device information field is OPTIONAL for the User Device both to implement as well as use in a configuration.

User Device Vendor ID identifies a User Device product manufacturer. This field is 3 octets in length and its value is the IEEE OUI or CID of the User Device vendor. User Device Product ID is variable length field that identifies a product of a User Device vendor. The User Device Product ID is assigned by the User Device vendor. User Device Firmware Version is variable length field that identifies the software firmware version executing on the User Device.

10.2.2 CONTROL FLOW

The CONTROL FLOW command SHALL be used by the Reader to indicate a transaction failure only when the secure channel has not yet been/could not be established. The command MAY be used

at the end of either expedited-standard, expedited-fast or step-up phases. Given the unprotected nature of the CONTROL FLOW command, the returned errors are intentionally non-descriptive.

10.2.2.1 Command Message

The INS, P1 and P2 values of the CONTROL FLOW command message SHALL be coded according to the following table.

Table 10-5 – CONTROL FLOW command header

Code	Value
INS	0x3C
P1	0x00
P2	0x00

The command data field of the CONTROL FLOW command message SHALL be coded according to the following table.

The CONTROL FLOW command data field length SHALL NOT exceed 255 Bytes.

Table 10-6 – CONTROL FLOW command data field

Tag	Length (Octets)	Description	Field is
0x41	1	S1_parameter, see below	mandatory
0x42	1	S2_parameter, see below	mandatory
0x63	variable	The content of Reader Descriptor TLV object (including the Tag 0xB5 byte and its length field), see Table 8-17	optional

The S1 parameter SHALL be coded according to the following values:

- 0x00 transaction finished with failure.

The S2 parameter SHALL be coded according to the following values:

- 0x00 no information.
- 0x27 protocol version not supported.

Other values for S1 and S2 are reserved for future use.

10.2.2.2 Response Message

The User Device SHALL send the CONTROL FLOW response message with an empty response data field.

11 Bluetooth LE Interface

The Bluetooth LE channel is required to establish and manage the UWB secure ranging service.

11.1 User Aliro Flows

The User Device and Reader MAY perform Bluetooth LE + UWB Aliro flow in which the Bluetooth LE SHALL be used as the transport protocol while UWB SHALL be used for securely determining proximity between the User Device and the Reader.

The User Device and Reader MAY perform Bluetooth-LE only Aliro flow in which Bluetooth LE SHALL be used as transport protocol and user SHALL explicitly select Reader for access action such as unlock, etc.

11.1.1 Bluetooth LE + UWB Aliro Flow

All Aliro messages in the Bluetooth LE + UWB Aliro flow SHALL follow the Aliro message rules in section 11.9. Figure 11-1 is an informative depiction of the Bluetooth LE + UWB Aliro flow.

The Bluetooth LE + UWB Aliro flow begins with Bluetooth LE discovery and L2CAP connection-oriented channel establishment (see sections 11.4 and 11.5) between the User Device and the Reader. Then the User Device SHALL send Initiate Access Protocol Message ID (see section 11.7.3.7) in clear over the unencrypted L2CAP connection to the Reader to trigger Reader for initiating Access Protocol.

The Reader SHALL then initiate the Access Protocol starting with the AUTH0 command to initiate either expedited-standard phase or expedited-fast phase on receipt of Initiate Access Protocol Message ID. The Reader SHALL send Event Message ID carrying General Error Attribute ID in clear to the User Device, if it is unable to initiate expedited-standard or expedited-fast phase on receipt of Initiate Access Protocol Message ID.

If the cryptogram verification in the expedited-fast phase fails at the Reader, the Reader SHALL continue with the expedited-standard phase, except it MAY for security reasons abort the transaction by sending Event Message ID carrying General Error Attribute ID to the User Device.

The Reader SHALL send Event Message ID carrying General Error Attribute ID to the User Device, if the signature verification in expedited-standard phase fails at the Reader.

The Reader SHALL send EXCHANGE command including tag 0x98 (see section 8.3.3.5.1) to prompt the User Device to make URSK available to the UWB ranging sensor (see section 8.3.3.5), if the cryptogram verification in the expedited-fast phase or the signature verification in the expedited-standard phase is successful at the Reader. The User Device SHALL send EXCHANGE response indicating the success or failure status of EXCHANGE processing. The behavior of the User Device and the Reader is described in section 8.3.3.5, in case error codes are reported in EXCHANGE response. Additional EXCHANGE commands MAY be sent.

If the User Device receives more than one tag 0x98 as part of the EXCHANGE command in a single session, an error is returned.

The Reader MAY initiate step-up phase only if successful processing is indicated in the EXCHANGE response by the User Device. The Reader SHALL indicate completion of Access Protocol and its status to the User Device with Reader Status Access Protocol Completed Message

ID carrying Reader Information Attribute ID after successful completion of step-up phase. In case the Reader does not initiate step-up phase then the Reader SHALL indicate completion of Access Protocol and its status to the User Device with Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID, if successful processing is indicated in EXCHANGE response by the User Device.

The Reader MAY send EXCHANGE commands according to section 8.3.3.5. If the Reader sends an EXCHANGE command, the User Device SHALL send EXCHANGE response indicating the success or failure status of EXCHANGE processing. The behavior of the User Device and the Reader is described in section 8.3.3.5, in case error codes are reported in EXCHANGE response.

After sending Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID, the Reader SHALL delete ExpeditedSKReader, delete StepUpSKReader if available, and the Reader SHALL NOT send any AP_RQ Message IDs to the User Device over this L2CAP connection. Likewise, after receiving Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID, the User Device SHALL delete ExpeditedSKDevice, delete StepUpSKDevice if available, and the User Device SHALL NOT send Initiate Access Protocol Message ID and any AP_RS Message IDs over this L2CAP connection.

If the User Device does not receive tag 0x98 in EXCHANGE command during Expedited phase in a session, then it SHALL send Event Message ID with General Error Attribute ID indicating ‘URSK_Unavailable’ upon receiving Reader Status Access Protocol Completed Message ID.

If the Reader indicates the support of Time Synchronization Procedure 0 feature inside the Reader SPSM and ALIRO BLE UWB Protocol Version Characteristic (see Table 11-7 and Table 11-9), the User Device SHALL send Time Sync Message ID (see section 11.7.4.2) according to Procedure 0 (see section 19.4 in [2]) immediately after sending Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID. If the Device UWB Clock is ‘Not in sync’ while Bluetooth LE connected with the User Device and the Reader supports the Time Synchronization Procedure 1 feature, the Reader SHALL trigger the Procedure 1, if all conditions are met (see section 19.4 in [2]).

The User Device or the Reader MAY Initiate UWB ranging session setup. If the Reader indicates the support of at least one Time Synchronization Procedure, at least one Time Sync Message ID SHALL be transmitted and received, respectively, before initiating the UWB ranging setup. When User Device initiates UWB ranging session setup, it SHALL trigger the Reader to initiate UWB ranging session setup by sending Ranging Message ID carrying Initiate Ranging Session Attribute ID. The Reader SHALL respond with Event Message ID carrying General Error Attribute ID, if it is unable to initiate UWB ranging session setup. Otherwise, the Reader starts UWB ranging session setup by sending Ranging Session Setup M1 Message ID (see section 12.1.4).

When the Reader initiates UWB ranging session setup, it SHALL send Ranging Session Setup M1 Message ID without receiving Ranging Message ID carrying Initiate Ranging Session Attribute ID from the User Device. The User Device SHALL respond with Ranging Message ID carrying Initiate Ranging Session Setup Later Attribute ID, if it cannot do UWB ranging session setup at this time. The Reader SHALL resend Ranging Session Setup M1 Message ID only after receiving Ranging Message ID carrying Initiate Ranging Session Attribute ID. The User Device responds with Ranging Session Setup M2 Message ID, if it can do UWB ranging session setup at this time.

The proximity measurement using secure UWB ranging over UWB radio is described in section 12. An ongoing UWB ranging session can be suspended and resumed according to section 11.1.1.2.

11.1.1.1 Reader status reporting

The Reader decision to change its status is out of scope of this specification. However, whenever there is a change in the Reader's status information, the Reader sends Reader Status Changed Message ID with a State Attribute ID (see section 11.7.3.3.1) according to its capabilities indicated in Reader Status Access Protocol Completed Message ID (see section 11.7.3.4), indicating the new Reader state.

If the Reader knows its state, it SHALL send this message with State Attribute ID indicating, 'Reader has started the operation to enter the secure state', 'Reader has started the operation to enter the unsecure state', or 'Reader state is jammed', whenever it transitions between Reader's states. Once in secure or unsecure Reader states, the Reader MAY send this message with a State Attribute ID indicating 'secure' or 'unsecure' to reflect the current Reader state.

If the Reader does not know its state, it SHALL send this message with State Attribute ID indicating, 'Reader state is not available at the Reader'.

The Reader SHALL NOT send duplicate Reader Status Changed Message ID messages to a BLE connected User Device.

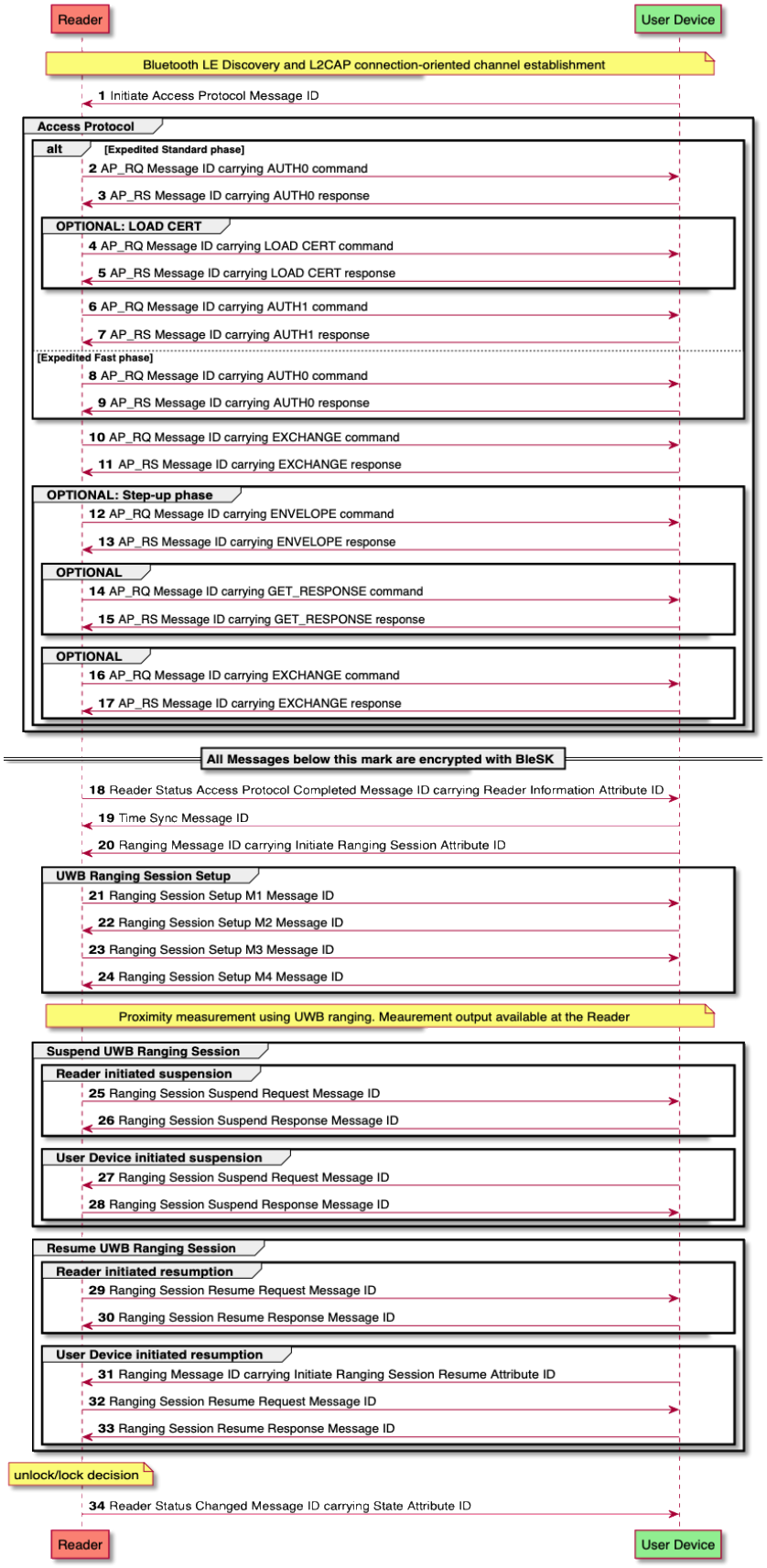

Reader
User Device
Bluetooth LE Discovery and L2CAP connection-oriented channel establishment
1 Initiate Access Protocol Message ID
Access Protocol
alt
[Expedited Standard phase]
2 AP_RQ Message ID carrying AUTH0 command
3 AP_RS Message ID carrying AUTH0 response
OPTIONAL: LOAD CERT
4 AP_RQ Message ID carrying LOAD CERT command
5 AP_RS Message ID carrying LOAD CERT response
6 AP_RQ Message ID carrying AUTH1 command
7 AP_RS Message ID carrying AUTH1 response
[Expedited Fast phase]
8 AP_RQ Message ID carrying AUTH0 command
9 AP_RS Message ID carrying AUTH0 response
10 AP_RQ Message ID carrying EXCHANGE command
11 AP_RS Message ID carrying EXCHANGE response
OPTIONAL: Step-up phase
12 AP_RQ Message ID carrying ENVELOPE command
13 AP_RS Message ID carrying ENVELOPE response
OPTIONAL
14 AP_RQ Message ID carrying GET_RESPONSE command
15 AP_RS Message ID carrying GET_RESPONSE response
OPTIONAL
16 AP_RQ Message ID carrying EXCHANGE command
17 AP_RS Message ID carrying EXCHANGE response
All Messages below this mark are encrypted with BleSK
18 Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID
19 Time Sync Message ID
20 Ranging Message ID carrying Initiate Ranging Session Attribute ID
UWB Ranging Session Setup
21 Ranging Session Setup M1 Message ID
22 Ranging Session Setup M2 Message ID
23 Ranging Session Setup M3 Message ID
24 Ranging Session Setup M4 Message ID
Proximity measurement using UWB ranging. Meaurement output available at the Reader
Suspend UWB Ranging Session
Reader initiated suspension
25 Ranging Session Suspend Request Message ID
26 Ranging Session Suspend Response Message ID
User Device initiated suspension
27 Ranging Session Suspend Request Message ID
28 Ranging Session Suspend Response Message ID
Resume UWB Ranging Session
Reader initiated resumption
29 Ranging Session Resume Request Message ID
30 Ranging Session Resume Response Message ID
User Device initiated resumption
31 Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID
32 Ranging Session Resume Request Message ID
33 Ranging Session Resume Response Message ID
unlock/lock decision
34 Reader Status Changed Message ID carrying State Attribute ID
Reader
User Device

Figure 11-1 – Bluetooth LE + UWB Aliro flow

11.1.1.2 Suspend and Resume UWB Ranging Session

Once a UWB ranging session is established, it can be active until the URSK is discarded (see section 11.1.1.4). However, for power optimizations, either the Reader or the User Device can suspend an active ranging session.

The Reader or the User Device MAY request to suspend the current ranging session by sending Ranging Session Suspend Request Message ID (see section 11.7.2.6). The User Device or the Reader respond with Ranging Session Suspend Response Message ID (see section 11.7.2.7) that indicates the status of the suspend request. The User Device and the Reader SHALL suspend the current UWB ranging session only if 'Request Accepted' status is indicated in the Ranging Session Suspend Response Message ID. Otherwise, the current UWB ranging session is not suspended. Furthermore, the Reader or the User Device MAY unilaterally suspend the current ranging session by sending Ranging Message ID carrying Ranging Session Suspended Attribute ID (see section 11.7.3.2.6). The User Device or Reader SHALL suspend the current ranging session on receiving Ranging Message ID carrying Ranging Session Suspend Attribute ID.

The Reader MAY resume a suspended UWB ranging session by sending Ranging Session Resume Request Message ID (see section 11.7.2.8) to the User Device. The User Device MAY trigger the UWB ranging session resumption by sending Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID (see section 11.7.3.2.2) to the Reader. Before initiating UWB ranging session resumption, the Reader SHALL confirm that there exists a non-expired URSK associated with the suspended UWB ranging session that is intended for resumption.

When the User Device receives the Ranging Session Resume Request Message ID, it SHALL identify the same set of configurations used to establish the UWB ranging session for the provided UWB Session Identifier. The User Device SHALL indicate a new UWB Time0 and STS Index0 in the Ranging Session Resume Response Message ID. The User Device MAY send Ranging Message ID carrying Initiate Ranging Session Resume Later Attribute ID, if the User Device is unable to resume an existing UWB ranging session on receipt of Ranging Session Resume Request Message ID. The Reader SHALL resend Ranging Session Resume Request Message ID only after it receives Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID.

11.1.1.3 Secure Ranging over UWB Radio Failed Handling

The Reader can fail to establish secure ranging over the UWB radio even after successfully receiving Ranging Session Setup M4 Message ID. For example, this occurs when the Reader does not receive any UWB ranging packets over the UWB radio. The Reader and the User Device can remedy this problem by redoing UWB ranging session setup and reusing the UWB Session Identifier and the URSK of the preceding UWB ranging session setup between themselves if the URSK (see section 11.1.1.4) is not discarded.

The Reader SHALL send Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attribute ID, if it failed to establish secure ranging over UWB radio after receiving Ranging Session Setup M4 Message ID or Ranging Session Resume Response Message ID from the User Device.

On receiving Ranging Message ID carrying Secure Ranging over UWB Radio Failed Attribute ID, the User Device MAY send Ranging Message ID carrying Initiate Ranging Session Attribute ID to the Reader. If the Reader responds with Ranging Session Setup M1 Message ID then it SHALL reuse the UWB Session Identifier and the URSK of the preceding UWB ranging session setup between the User Device and the Reader, if the URSK is not discarded.

11.1.1.4 URSK Lifetime

The URSK SHALL be discarded if one of the following conditions is met:

1. The STS_INDEX reaches its maximum value of 2^31-1.
2. The STS_INDEX is lost, and it cannot be ensured that a previously used STS_INDEX will not be used again.
3. URSK time to live has expired. The URSK time to live is set to 12 hours in this specification. The URSK time to live countdown starts when the first dURSK is derived. The first dURSK SHALL be derived immediately before sending or after receiving Ranging Session Setup M4 Message ID for the User Device and the Reader, respectively [2].
4. The Bluetooth LE connection between the User Device and the Reader is terminated or not available.

The User Device or the Reader SHALL send Event Message ID carrying General Error Attribute ID with reason code URSK_Unavailable, if URSK is discarded at the User Device or the Reader due to reasons 1, 2 or 3 above.

11.1.2 Bluetooth LE-Only Aliro Flow

All the Aliro messages in the Bluetooth LE-Only Aliro flow SHALL follow the Aliro message rules in section 11.9. Figure 11-2 is an informative depiction of Bluetooth LE-only Aliro flow.

The Bluetooth LE-only Access flow begins with Bluetooth LE discovery and L2CAP connection-oriented channel establishment (see sections 11.4 and 11.5) between the User Device and the Reader. After the intent to request Bluetooth LE-only Access flow is determined, the User Device SHALL send Initiate Access Protocol RKE Message ID (see section 11.7.3.8) in clear over the unencrypted L2CAP connection to the Reader to trigger Reader for initiating Access Protocol.

The Reader SHALL then initiate the Access Protocol starting with the AUTH0 command to initiate expedited-standard phase. expedited-fast phase SHALL NOT be used in Bluetooth LE-only Aliro flow. The Reader SHALL send Event Message ID carrying General Error Attribute ID in clear to the User Device, if it is unable to initiate Expedited-standard phase on receipt of Initiate Access Protocol RKE Message ID. The Reader SHALL send Event Message ID carrying General Error Attribute ID in clear to the User Device, if the signature verification in expedited-standard phase fails at the Reader.

The Reader MAY send EXCHANGE commands according to section 8.3.3.5. If the Reader sends an EXCHANGE command, the User Device SHALL send EXCHANGE response indicating the success or failure status of EXCHANGE processing. The behavior of the User Device and the Reader is described in section 8.3.3.5, in case error codes are reported in EXCHANGE response.

The Reader MAY initiate step-up phase, if the signature verification in the expedited-standard phase is successful at the Reader. The Reader SHALL indicate Access Protocol completion and its current state to the User Device with Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID after successful completion of step-up phase. In case the Reader does not

initiate step-up phase then the Reader SHALL indicate Access Protocol completion and its status to the User Device with Reader Access Protocol Completed Message ID carrying Reader Information Attribute ID, if the signature verification in the expedited-standard phase is successful at the Reader.

After sending the SessionData messages as part of the step-up phase, the Reader MAY send EXCHANGE commands according to section 8.3.3.5. If the Reader sends an EXCHANGE command, the User Device SHALL send EXCHANGE response indicating the success or failure status of EXCHANGE processing. The behavior of the User Device and the Reader is described in section 8.3.3.5, in case error codes are reported in EXCHANGE response.

The Reader SHALL delete ExpeditedSKReader, delete StepUpSKReader if available, after sending Reader Status Access Protocol Completed Message ID carrying State Attribute ID. The User Device SHALL delete ExpeditedSKDevice, delete StepUpSKDevice if available, after receiving Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID. The User Device SHALL send RKE Request Message ID carrying user registered intent as Action Attribute ID. The Reader Status reporting in Bluetooth LE-Only Aliro Flow is described in section 11.1.1.1.

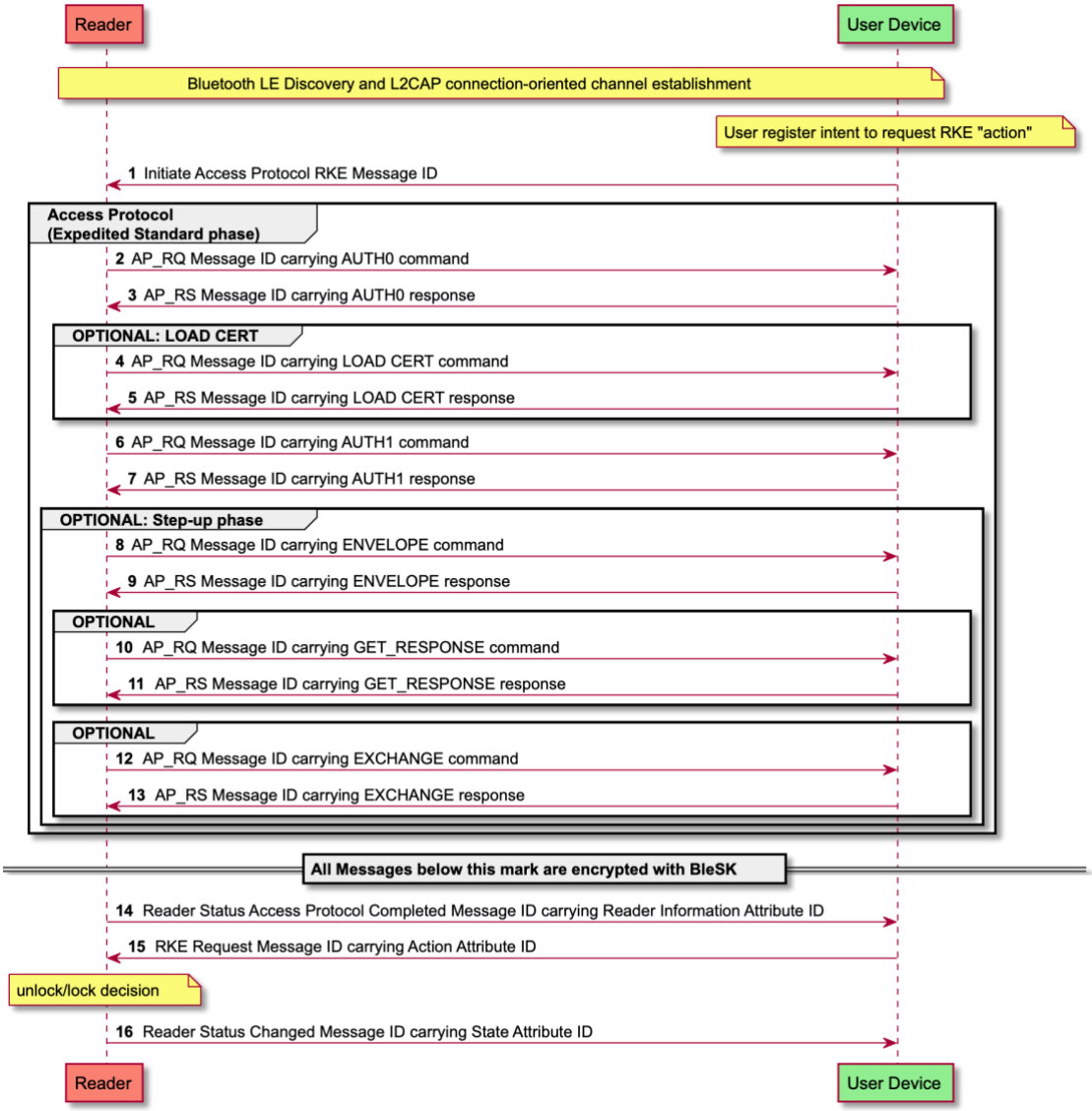


Figure 11-2 – Bluetooth LE-Only Aliro flow

11.1.3 Encryption and Authentication in Aliro Flows

The Aliro messages between the User Device and the Reader for Bluetooth LE + UWB Aliro flow and the Bluetooth LE-Only Aliro flow are encrypted and authenticated (see section 11.8).

The Event Message ID carrying General Error Attribute ID and Event Message ID carrying Busy Attribute ID SHALL be sent in clear in Bluetooth LE + UWB Aliro flow and Bluetooth LE-Only Access flow by the User Device and the Reader until the Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID is received and transmitted by the User Device and the Reader, respectively. When Event Message ID carrying General Error Attribute ID is sent in clear only Unknown Error reason code is allowed to be indicated (see section 11.7.3.1.2).

All Aliro messages SHALL be sent encrypted and authenticated according to section 11.8 in Bluetooth LE + UWB Aliro flow and Bluetooth LE-Only Aliro flow by the User Device and the Reader after (and including) the Reader Status Access Protocol Completed Message ID carrying

Reader Information Attribute ID is received and transmitted by the User Device and the Reader, respectively.

11.2 Bluetooth LE Requirements

The Bluetooth LE Controller and Host for both the User Device and Reader SHALL be compliant with the mandatory capabilities of the Bluetooth Core Specification 4.2 or later. To future proof Bluetooth LE operation in this Aliro specification, it is recommended that both the User Device and the Reader comply with the mandatory capabilities of the Bluetooth Core Specification 5.2 or later.

The Bluetooth LE pairing between the User Device and the Reader is OPTIONAL and not required for this Aliro specification.

11.2.1 Reader

The Reader SHALL support GAP Peripheral role.

The Reader SHALL support GATT in the server role.

11.2.2 User Device

The User Device SHALL support GAP Central role.

The User Device SHALL support GATT in the client role.

11.3 Bluetooth LE Advertising

The Reader SHALL advertise on the LE 1M PHY. When advertising on the LE 1M PHY, the connectable and scannable undirected event SHALL be used, see Vol 6, PART B, section 4.4.2.7 of [1].

The Reader MAY advertise on both the LE Coded PHY with S = 2 coding and on the LE 1M PHY. If both PHY are used, the advertising for each SHALL occur consecutively. When advertising on the LE Coded PHY, the connectable and undirected event SHALL be used, see Vol 6, Part B, section 4.4.2.3 in [1].

The ADV_IND contains Advertising Address and Advertising Data (AD) shown in Table 11-1 and Table 11-2, respectively.

Table 11-1 – AdvA field of ADV_IND

Field	Length (Octets)	Value	Description
Advertising Address (AdvA)	6	variable	Static Address as defined in [1]

Table 11-2 – Payload of ADV_IND

Byte	Value	Description
0	0x02	AD[0] Length == 2 Bytes
1	0x01	AD[0] Type == 1 (Flags)
2	0x06	Bit 0 (LE Limited Discoverable Mode) SHOULD be set to 0 Bit 1 (LE General Discoverable Mode) SHOULD be set to 1 If only Bluetooth LE is supported, this value SHOULD be set to 0x06. If BR/EDR functionality is supported by the Reader, this value SHOULD be set accordingly.
3	0x1B	AD[1] Length == 27
4	0x16	AD[1] Type == 0x16 (Service Data – 16-bit UUID [1])
5-6	0xFFF2	16-bit Aliro service UUID assigned by BT SIG
7	Variable	Bit 7: Bluetooth LE + UWB Aliro flow supported. Bit 6: Bluetooth LE-Only Aliro flow supported. Bit 5 is RFU. Bits [4:3]: Notification Bits [2:0]: Aliro Bluetooth LE Advertisement Version
8	Variable	Tx Power Level in dBm
9-16	Variable	truncated_reader_group_identifier
17-18	Variable	truncated_reader_group_sub_identifier
19-22	Variable	Dynamic Tag Expiry Timestamp
23	0x00	RFU
24 – 30	Variable	Dynamic Tag

The Aliro service SHALL be identified by the 0xFFF2 16-bit UUID that is assigned by Bluetooth SIG.

The Bluetooth LE + UWB Aliro flow supported bit (bit 7 in byte 7 in Table 11-2) is set to 01_h, if the Reader supports Bluetooth LE + UWB Aliro flow (see section 11.1.1). Otherwise, the Reader does not support Bluetooth LE + UWB Aliro flow.

The Bluetooth LE-Only Aliro flow supported bit (bit 6 in byte 7 in Table 11-2) is set to 01_h, if the Reader supports Bluetooth LE-Only Aliro flow (see section 11.1.2). Otherwise, the Reader does not support Bluetooth LE-Only Aliro flow.

Notification field indicates information about the Reader. The values 0, 1, 2, 3 indicate No Error, Unknown Error, Low Battery, and Sensor Triggered respectively. The Reader MAY indicate Sensor Triggered, if it determines some physical interaction occurred at the Reader, such as Reader is physically touched. The Reader indicates Sensor Triggered in at least 10 Bluetooth LE advertisements it sends after determining the occurrence of the physical interaction.

Aliro Bluetooth LE Advertisement Version field indicates version of Aliro Bluetooth LE advertisement. The Aliro Bluetooth LE Advertisement Version is changed whenever there is a change to the payload of Aliro Bluetooth LE advertisement. In this specification, Aliro Bluetooth LE Advertisement Version is set to 0. Values 1 – 7 are RFU, respectively.

The Tx Power Level in dBm Data field is the current radiated transmit power of a Bluetooth LE module. The radiated power is sum integrated over all directions (section 3.237 in [16]). The allowed range of values is [-100,20]. Other values are RFU. Note that the Tx Power of Bluetooth LE module SHALL be the same between the Bluetooth LE advertisement and the Bluetooth LE Connection.

truncated_reader_group_identifier field is the first 8 octets of the reader_group_identifier (see section 6.2). truncated_reader_group_sub_identifier is the first 2 octets of the reader_group_sub_identifier (see section 6.2).

Dynamic Tag Expiry Timestamp field is the Unix timestamp represented by a 32-bit unsigned integer. This field indicates expiration time of the Dynamic Tag field. If current Unix timestamp at the Reader is unavailable, then Dynamic Tag Expiry Timestamp is set to 0xFFFFFFFF.

Dynamic Tag field carries the dynamic tag data generated as in section 11.3.1.

The Dynamic Tag Expiry Timestamp Data field remains unchanged until its expiry. At the expiry of the Dynamic Tag field, the Reader generates new values of Dynamic Tag Expiry Timestamp and Dynamic Tag and advertises them.

The AdvA and service UUID field are transmitted with least significant byte first as per [1], other multi-octet fields in the advertisement packet are transmitted with most significant byte first.

11.3.1 Dynamic Tag Generation at the Reader

Security function *e* generates 128-bit encryptedData from a 128-bit Group Resolving Key and 128-bit plaintextData using AES-128-bit block cypher as defined in [21]

encryptedData = *e*(Group Resolving Key, plaintextData), where plaintextData = Pad_Bytes || AdvA || Dynamic Tag Expiry Timestamp.

The Pad_Bytes are set to 0x000000000000.

The most significant octet of Group Resolving Key corresponds to key[0], the most significant octet of plaintextData corresponds to in[0] and most significant octet of encryptedData corresponds to out[0] using the notation specified in FIPS-197 [21]. Note the security function *e* is used in [1]. The AdvA and Dynamic Tag Expiry Timestamp are formatted with most significant byte first in the plaintextData field.

Dynamic Tag is the 7 most significant octets of the encryptedData.

11.3.2 Bluetooth LE Parameter Configuration Example

The Reader sets the advertising interval on the LE 1M PHY and LE Coded PHY with S = 2 coding. The User Device selects the connection interval when connecting to the Reader.

An example parameter configuration by this specification is below.

- Connection interval at the User Device = 30 ms
- Advertising interval on LE 1M PHY at the Reader = 42.5 ms
- Advertising interval on LE Coded PHY with S = 2 at the Reader = 84 ms.

11.4 Bluetooth LE Link Layer Connection Establishment

The Reader begins by sending ADV_IND with Aliro service UUID and the advertising payload in section 11.3. The Reader SHALL be in the advertising state with its filtering policy set to accept all connection requests.

The User Device begins passive scanning. The User Device filter policy to identify Reader of interest is out of scope of the specification.

An example filter policy can use Aliro service UUID along with advertisement address and/or truncated_reader_group_identifier and/or truncated_reader_group_sub_identifier and/or Dynamic Tag (see section 11.4.1).

The User Device SHALL send a connection request (CONNECT_IND) to the Reader of interest.

11.4.1 Example of Dynamic Tag-based filtering at the User Device

For each received advertisement containing Aliro service UUID, verify using Dynamic Tag Expiry Timestamp that the Dynamic Tag is not expired. Then, for each Group Resolving Key on the User Device, compute Dynamic Tag using Dynamic Tag Expiry Timestamp and advertisement address as described in section 11.3.1. If Dynamic Tag value in the received advertisement matches the computed Dynamic Tag, then a Reader of interest is identified.

11.5 Bluetooth LE GATT Flow

The User Device Host (acting as the GATT client) initiates service discovery with the Reader Host (acting as the GATT server) to get the BLE AC Service *UUID_SPSM_ALIRO_BLE_UWB_PROTOCOL_VERSION* characteristic to establish the L2CAP connection for the Aliro Service.

Figure 11-3 – L2CAP Connection-Oriented Channel is an example in which the User Device reads the Simplified Protocol / Service Multiplexer (SPSM) value from the Reader's GATT server. The User Device MAY read SPSM value in other ways [1]. For more details on the Box A in Figure 1, please refer to [1].

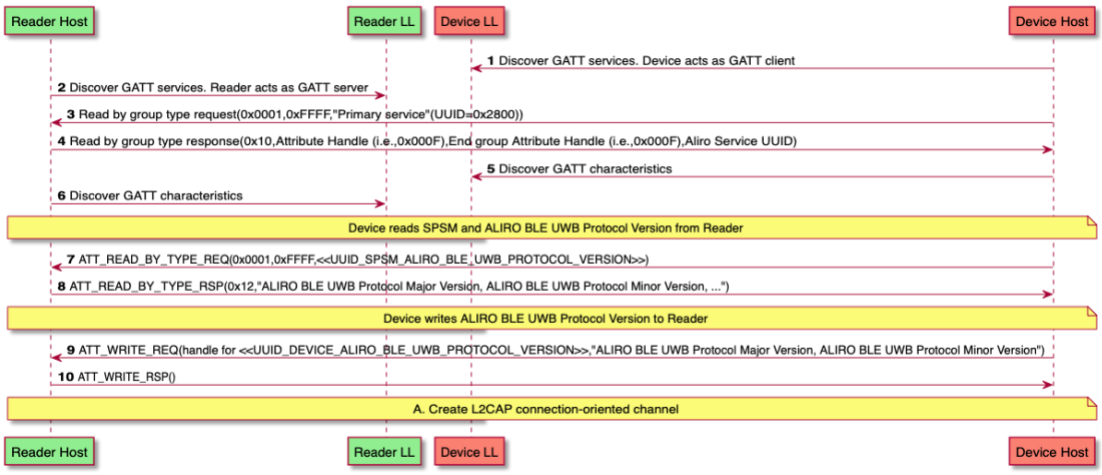


Figure 11-3 – L2CAP Connection-Oriented Channel

To prevent Aliro BLE UWB Protocol Version downgrade attacks, the list of Aliro BLE UWB Protocol Versions supported by the Reader and the ALIRO BLE UWB Protocol Version selected by the User Device are injected as inputs to the application-level secure channel key derivation defined in section 11.8.1. If either of these elements are tampered during transit over-the-air, the parties will obtain different session keys, and the tampering will be detected.

11.5.1 Reader PSM Characteristic

This characteristic SHALL return the SPSM used for the L2CAP channel from the Reader.

The User Device SHALL read the SPSM from the Reader upon connection. The Reader manufacturer SHALL pick an SPSM. For LE Credit-based Connections, a dynamically allocated SPSM value (i.e., the non-SIG assigned SPSM value in 0x0080-0x00FF) is used.

Upon Bluetooth LE connection, the User Device SHALL

- Read the characteristic by UUID with the UUID_SPSM_ALIRO_BLE_UWB_PROTOCOL_VERSION (see Table 11-3 and Table 11-4). The SPSM value uses big-endian byte order.
- Write the characteristics by UUID for UUID_DEVICE_ALIRO_BLE_UWB_PROTOCOL_VERSION (see Table 11-5 and Table 11-6).

11.5.2 ALIRO BLE UWB Protocol Version Characteristic

Table 11-3 – Reader SPSM and ALIRO BLE UWB Protocol Version Characteristic declaration

Attribute Handle	Attribute Type	Attribute Value			Attribute permission
	0x2803 – UUID for	Charac teristic	Characteri stic Value	Characteristic UUID =	Read only,

Attribute Handle	Attribute Type	Attribute Value			Attribute permission
	<<Characteristics>>	Properties = 0x02	Attribute Handle =0xMMMM	D3B5A130-9E23-4B3A-8BE4-6B1EE5F980A3-UUID_SPSM_ALIRO_BLE_UWB_PROTOCOL_VERSION	No Authentication No Encryption, No Authorization

Table 11-4 – Reader SPSM and ALIRO BLE UWB Protocol Version Characteristic Value declaration

Attribute Handle	Attribute Type	Attribute Value	Attribute permission
0xMMMM	D3B5A130-9E23-4B3A-8BE4-6B1EE5F980A3 – UUID_SPSM_ALIRO_BLE_UWB_PROTOCOL_VERSION	Refer to Table 11-7	Read only, No Authentication No Encryption

Table 11-5 – User Device Selected ALIRO BLE UWB Protocol Version Characteristic declaration

Attribute Handle	Attribute Type	Attribute Value			Attribute permission
0xNNNN	0x2803 – UUID for <<Characteristics>>	Characteristic Properties = 0x08	Characteristic Value Attribute Handle = 0xMMMM	Characteristic UUID = BD4B9502-3F54-11EC-B919-0242AC120005-UUID_DEVICE_ALIRO_BLE_UWB_PROTOCOL_VERSION	Read only, No Authentication No Encryption

Attribute Handle	Attribute Type	Attribute Value			Attribute permission
					No Authorization

Table 11-6 – User Device Selected ALIRO BLE UWB Protocol Version Characteristic Value declaration

Attribute Handle	Attribute Type	Attribute Value	Attribute permission
0xMMMM	BD4B9502-3F54-11EC-B919-0242AC120005-UUID_DEVICE_ALIRO_BLE_UWB_PROTOCOL_VERSION	Refer to Table 11-7 Table 11-8	Write only, No Authentication No Encryption, No Authorization

Table 11-7 – Attribute Value definition for Reader SPSM and ALIRO BLE UWB Protocol Version Characteristic

Attribute Value	Length (Octets)	Description
SPSM Value	2	SPSM value in big-endian byte order
Supported ALIRO BLE UWB Protocol Version Len	1	Length in octets for supported ALIRO BLE UWB Protocol version field
Supported ALIRO BLE UWB Protocol Version	2x n	ALIRO BLE UWB Protocol Version is 2 Octet field (Major: Minor). Minor version octet changes whenever there is a change in any of the following: Bluetooth Aliro message format (see section 11.7), Bluetooth LE + UWB Aliro

Attribute Value	Length (Octets)	Description
		flow (see section 11.1.1), Bluetooth LE-Only Aliro flow (see section 11.1.2). Major version octet changes when backward compatibility is broken. The Reader SHALL return all supported ALIRO BLE UWB Protocol Version using two octets per version in big-endian encoding. n is the number of ALIRO BLE UWB Protocol versions supported by the Reader. The ALIRO BLE UWB Protocol Version SHALL be ordered from the highest to the lowest version. In this specification the Reader SHALL support the ALIRO BLE UWB Protocol Version 1.0 (coded 0x0100). The User Device SHALL select the highest ALIRO BLE UWB Protocol Version commonly supported by the User Device and the Reader as described in Selected ALIRO BLE UWB Protocol Version attribute (see Table 11-8). The first supported ALIRO Message Version is 0x0100.
Features Supported Length	1	Indicates the length in bytes of the Features Supported field.
Features Supported	k	k is the length in bytes needed to carry the bitmap for the features supported by the Reader (see Table 11-9). This field is coded in big endian order.

Table 11-8 – Attribute Value definition for Selected ALIRO BLE UWB Protocol Version

Attribute Value	Length (Octets)	Description
Selected ALIRO BLE UWB Protocol Version	2	User Device-selected highest ALIRO BLE UWB Protocol Version commonly supported by the User Device and the Reader. In this specification, the User Device SHALL support ALIRO BLE UWB

Attribute Value	Length (Octets)	Description
		Protocol Version 1.0 (coded 0x0100).
Features Supported Length	1	Indicates the length in bytes of the Features Supported field.
Features Supported	k	k is the length in bytes needed to carry the bitmap for the features supported by the User Device (see Table 11-9). This field is coded in big endian order.

Table 11-9 Supported Features bitmap

Bits	Feature Name	Description
Bit0	Time Synchronization Procedure 0	Set to 1 when Time Synchronization Procedure 0 [2] is supported. Otherwise, Time Synchronization Procedure 0 is not supported.
Bit1	Time Synchronization Procedure 1	Set to 1 when Time Synchronization Procedure 1 [2] is supported. Otherwise, Time Synchronization Procedure 1 is not supported.
Bit2	LE Coded PHY	Set to 1 when LE Coded PHY is supported. Otherwise, LE Coded PHY is not supported.

11.6 Bluetooth LE Connection Teardown

Terminating an L2CAP channel requires that an L2CAP_DISCONNECTION_REQ be sent and acknowledged by an L2CAP_DISCONNECTION_RSP [1]. L2CAP channel teardown can be initiated by the User Device or the Reader.

11.7 Aliro Message Format

The Aliro messages SHALL be exchanged over the Bluetooth LE L2CAP using the Aliro Service SPSM. It is allowed to include one or more Aliro messages within a single L2CAP service data unit.

The User Device SHALL retrieve the Aliro Service SPSM from the Reader's GATT server and establish an LE L2CAP credit-based connection to the Reader's SPSM for use by the Ranging service.

Table 11-10 shows the Aliro message format. All multi-octet integer fields in Aliro message are encoded in big-endian order.

Table 11-10 – Aliro message format

Field	Length (Octets)	Value	Description
Protocol Header	1	See Table 11-11	Bits B5:B0 indicate Protocol Type. Bits B7:B6 are RFU and set to 0.
Message ID	1	See Table 11-11	Identifier of the Message.
Length	2	Variable	Indicates length in octets of the Payload field.
Payload	Variable	Variable	Contains data corresponding to the Protocol Type and Message ID.

When Protocol Type is equal to AP then the Payload field contains the commands listed in section 8.3 and 8.4.

When Protocol Type is equal to UWB Ranging Service and Supplementary Service then the Payload field carries one or more Attributes IDs (see sections 11.7.2 and 11.7.4, respectively). When the Protocol Type is equal to Notification then the Payload field SHALL carry only one Attribute ID (see section 11.7.3) to avoid presence of conflicting Attributes in one Notification. This rule has an exception: if the Payload field includes the General Error Attribute ID, the Reader Descriptor Attribute ID MAY also be included. For any other Attribute ID in the Payload field, only one Attribute ID is permitted, for Notification Protocol Type. When Protocol Type is equal to 3rd Party App Payload field content is defined in section 11.7.5.

When supported and configured for use, Reader Descriptor information SHALL be sent by the Reader once during an Access Protocol transaction over Bluetooth interface, in the first EXCHANGE or Event Message ID carrying Error Attribute ID and Reader Descriptor Attribute ID.

Zero length Payload field SHALL be treated as malformed Aliro message.

Table 11-11 – Protocol Type and Message ID in Aliro message

Protocol Type	Protocol Type Value	Message ID	Message ID Value
AP	0	AP_RQ	0
		AP_RS	1
		RFU	2 - 255
UWB Ranging Service	1	Ranging Session Setup M1	0
		Ranging Session Setup M2	1
		Ranging Session Setup M3	2
		Ranging Session Setup M4	3
		Ranging Session Suspend Request	4
		Ranging Session Suspend Response	5
		Ranging Session Resume Request	6
		Ranging Session Resume Response	7
		RFU	8 - 255
Notification	2	Event	0
		Ranging	1
		Reader Status Changed	2
		Reader Status Access Protocol Completed	3
		RKE Request	4
		Initiate Access Protocol	5
		Initiate Access Protocol RKE	6
		RFU	7 - 255
Supplementary Service	3	Time Sync	0
		RFU	1 - 255

Protocol Type	Protocol Type Value	Message ID	Message ID Value
3rd Party App	4	Pass Through	0
		RFU	1 - 255
RFU	5 - 63	-	-

Table 11-12 – Attribute format

Field	Length (Octets)	Value	Description
Attribute ID	1	Variable	Identifier of the Attribute.
Attribute Length	1	Variable	Indicates length in octets of the Attribute Value field.
Attribute Value	Variable	Variable	Contains data corresponding to the Attribute ID.

11.7.1 AP Protocol Type

11.7.1.1 AP_RQ Message ID

The Payload field contains the AP command.

11.7.1.2 AP_RS Message ID

The Payload field contains the AP command response.

11.7.2 UWB Ranging Service Protocol Type

11.7.2.1 Attribute IDs

The Attribute IDs for UWB Ranging Service Protocol Type are listed in Table 11-13.

Table 11-13 – Attribute IDs for UWB Ranging Service Protocol Type

Attribute ID	Attribute ID Value
UWB Configuration Identifier	0
Pulse Shape Combo	1
UWB Session Identifier	2

Attribute ID	Attribute ID Value
Channel Bitmask	3
RAN Multiplier	4
Slot Bitmask	5
SYNC Code Index Bitmask	6
SYNC Code Index	7
Hopping Configuration Bitmask	8
Number Chaps per Slot	9
Number Responders Nodes	10
Number Slots per Round	11
STS_Index0	12
UWB_Time0	13
HOP Mode Key	14
MAC Mode	15
Vendor Specific	16
Status	17
Reserved	18 – 255

11.7.2.1.1 UWB Configuration Identifier Attribute ID

The Attribute Length field is set to 2 x m, where m is the number of UWB Configuration Identifiers supported.

If the UWB Configuration Identifier Attribute ID is transmitted by the Reader, then Attribute Value field carries supported UWB Configuration Identifier value as described in Supported_UWB_Config_Id parameter in Ranging Capability Request in [2].

If the UWB Configuration Identifier Attribute ID is transmitted by the User Device, then Attribute Value field carries selected UWB Configuration Identifier value as described in Selected_UWB_Config_Id parameter in Ranging Capability Response in [2].

11.7.2.1.2 Pulse Shape Combo Attribute ID

The Attribute Length field is set to 1 x l, where l is the number of Pulse Shape Combos supported.

If Pulse Shape Combo Attribute ID is transmitted by the Reader, then Attribute Value field carries supported Pulse Shape Combo value as described in Supported_PulseShape_Combo parameter in Ranging Capability Request in [2].

If Pulse Shape Combo Attribute ID is transmitted by the User Device, then Attribute Value field carries selected Pulse Shape Combo value as described in Selected_PulseShape_Combo parameter in Ranging Capability Response in [2].

11.7.2.1.3 UWB Session Identifier Attribute ID

The Attribute Length field is set to four octets.

The Attribute Value field carries session identifier, which is the least significant four octets of the Transaction Identifier field (see Table 8-4).

11.7.2.1.4 Channel Bitmask Attribute ID

The Attribute Length field is set to one octet.

If Channel Bitmask Attribute ID is transmitted by the Reader, then the Attribute Value field carries Channel Bitmask value as described in Channel_Bitmask parameter in Ranging Session Request in [2].

If the Channel Bitmask Attribute ID is transmitted by the User Device, then the Attribute Value field carries Selected UWB Channel value as described in Selected_UWB_Channel parameter in Ranging Session Response in [2].

11.7.2.1.5 RAN Multiplier Attribute ID

The Attribute Length field is set to one octet.

If the RAN Multiplier Attribute ID is transmitted by the Reader, then the Attribute Value field carries the session RAN Multiplier value as described in Session_RAN_Multiplier parameter in Ranging Session Setup Request in [2].

If the RAN Multiplier Attribute ID is transmitted by the User Device, then the Attribute Value field carries RAN Multiplier value as described in RAN_Multiplier in Ranging Session Response in [2].

11.7.2.1.6 Slot Bitmask Attribute ID

The Attribute Length field is set to one octet.

The Slot Bitmask Attribute ID is transmitted by the User Device. The Attribute Value field carries Slot Bitmask value as described in Slot_BitMask parameter in Ranging Session Response in [2].

11.7.2.1.7 SYNC Code Index Bitmask Attribute ID

The Attribute Length field is set to four octets.

If the SYNC Code Index Bitmask Attribute ID is transmitted by the User Device, then Attribute Value field carries SYNC Code Index Bitmask value as described in SYNC_Code_Index_BitMask parameter in Ranging Session Response in [2].

If the SYNC Code Index Bitmask Attribute ID is transmitted by the Reader, then Attribute Value field carries SYNC Code Index Bitmask value as described in SYNC_Code_Index parameter in Ranging Session Setup Request in [2].

11.7.2.1.8 Sync Code Index Attribute ID

The Attribute Length field is set to one octet.

The Sync Code Index Attribute ID is transmitted by the User Device. The Attribute Value field carries the selected Sync Code Index value as described in SYNC_Code_Index parameter in Ranging Session Setup Response in [2].

11.7.2.1.9 Hopping Configuration Bitmask Attribute ID

The Attribute Length field is set to one octet.

If the Hopping Configuration Bitmask Attribute ID is transmitted by the User Device, then Attribute Value field carries hopping configuration bitmask value as described in Hopping_Config_Bitmask in Ranging Session Response in [2].

If the Hopping Configuration Bitmask Attribute ID is transmitted by the Reader, then Attribute Value field carries selected hopping configuration bitmask as described in Selected_Hopping_Config_Bitmask in Ranging Session Setup Request in [2].

In this specification, User Device and Reader shall support default hopping sequence (see section 17). The AES-based hopping sequence defined in [2] shall not be used in this specification.

11.7.2.1.10 Number Chaps per Slot Attribute ID

The Attribute Length field is set to one octet.

The Number Chaps per Slot Attribute ID is transmitted by the Reader. The Attribute Value field carries Number Chaps per Slot value as described in Number_Chaps_per_Slot parameter in Ranging Session Setup Request in [2].

11.7.2.1.11 Number Responders Nodes Attribute ID

The Attribute Length field is set to one octet.

The Number Responder Nodes Attribute ID is transmitted by the Reader. The Attribute Value field carries Number Responder Nodes value as described in Number_Responder_Nodes parameter in Ranging Session Setup Request in [2].

11.7.2.1.12 Number Slot per Round Attribute ID

The Attribute Length field is set to one octet.

The Number Slot per Round Attribute ID is transmitted by the Reader. The Attribute Value field carries Number Slot per Round value as described in Number_Slot_per_Round parameter in Ranging Session Setup Request in [2].

11.7.2.1.13 STS Index0 Attribute ID

The Attribute Length field is set to four octets.

The STS Index0 Attribute ID is transmitted by the User Device. The Attribute Value field carries STS Index0 value as described in STS_Index0 parameter in Ranging Session Setup Response in [2].

11.7.2.1.14 UWB Time0 Attribute ID

The Attribute Length field is set to eight octets.

The UWB Time0 Attribute ID is transmitted by the User Device. The Attribute Value field carries UWB Time0 value as described in UWB_Time0 parameter in Ranging Session Setup Response in [2].

11.7.2.1.15 HOP Mode Key Attribute ID

The Attribute Length field is set to four octets.

The HOP Mode Key Attribute ID is transmitted by the User Device. The Attribute Value field carries HOP Mode Key value as described in HOP_Mode_Key parameter in Ranging Session Setup Response in [2].

11.7.2.1.16 MAC Mode Attribute ID

The Attribute Length field is set to one octet.

The MAC Mode Attribute ID is transmitted by the Reader. The Attribute Value field carries bitmask:

[b7, b6] indicate number of ranging rounds out of all the ranging rounds in a ranging block that are used for UWB ranging procedure. The decimal values 0 and 1 for [b7, b6] corresponds to 1 and 2 ranging rounds out of all the ranging rounds in a ranging block are used for UWB ranging procedure, respectively. The decimal values 2 and 3 for [b7, b6] are reserved for future.

[b5, b0] indicates the offset between the two ranging rounds out of all the ranging rounds in a ranging block that are used for UWB ranging procedure. The value range is $1 \leq O^k \leq N^k_{Round} - 1$. Bits [b5, b0] SHALL be set if [b7, b6] set to decimal value 1. Otherwise, bits [b5, b0] are reserved for future use and set to 0.

11.7.2.1.17 Vendor Specific Attribute ID

Vendor Specific Attribute is carrying the information that is not defined in this standard.

The Attribute Length field is set to 3 + length of the Attribute Value octets.

The Attribute Length field is followed by an IEEE OUI or CID of 3 octets. The OUI or CID field is set to variable value defined in IEEE Registration Authority.

The Attribute Value field in Attribute carries vendor specific information.

11.7.2.1.18 Status Attribute ID

The Status Attribute ID is transmitted by the Reader and the User Device. The Attribute Length field is set to one octet. The Attribute Value field is in Table 11-14.

Table 11-14 – Status Attribute values

Value	Meaning
0	Request Accept
1	Request Reject

Value	Meaning
2-255	Reserved

11.7.2.2 Ranging Session Setup M1 Message ID

The Ranging Session Setup M1 Message ID is sent by the Reader to the User Device during ranging session setup exchange.

The following Attribute IDs are carried in the Payload field:

1. UWB Configuration Identifier
2. Pulse Shape Combination
3. Channel Bitmask
4. UWB Session Identifier
5. Vendor Specific attribute is optionally present.

11.7.2.3 Ranging Session Setup M2 Message ID

The Ranging Session Setup M2 Message ID is sent by the User Device to the Reader in response to Ranging Session Setup M1 Message ID during ranging session setup exchange.

The following Attribute IDs are carried in the Payload field:

1. UWB Configuration Identifier
2. Pulse Shape Combination
3. Channel Bitmask
4. SYNC Code Index Bitmask
5. RAN Multiplier
6. Slot Bitmask
7. Hopping Configuration Bitmask
8. Vendor Specific attribute is optionally present.

11.7.2.4 Ranging Session Setup M3 Message ID

This Ranging Session Setup M3 Message ID is sent by the Reader to the User Device in response to Ranging Session Setup M2 Message ID during ranging session setup exchange.

The following Attribute IDs are carried in the Payload field:

1. RAN Multiplier
2. Number Chaps per Slot
3. Number Responders Nodes
4. Number Slots per Round
5. SYNC Code Index Bitmask
6. Hopping Configuration Bitmask
7. MAC Mode

8. Vendor Specific attribute is optionally present.

11.7.2.5 Ranging Session Setup M4 Message ID

This Ranging Session Setup M4 Message ID is sent by the User Device to the Reader in response to a Ranging Session Setup M3 Message ID during ranging session setup exchange.

The following Attribute IDs are carried in the Payload field:

1. STS Index0
2. UWB Time0
3. HOP Mode Key
4. SYNC Code Index
5. Vendor Specific attribute is optionally present.

11.7.2.6 Ranging Session Suspend Request Message ID

This Ranging Session Suspend Request Message ID is used to request suspension of an active ranging session for a given UWB session identifier.

The following Attribute IDs are carried in the Payload field:

1. UWB Session Identifier

11.7.2.7 Ranging Session Suspend Response Message ID

This Ranging Session Suspend Response Message ID is a response to Ranging Session Suspend Request Message ID.

The following Attribute ID is carried in the Payload field:

1. Status

11.7.2.8 Ranging Session Resume Request Message ID

This Ranging Session Resume Request Message ID is used to request resumption of a suspended ranging session with a given UWB session identifier.

The following Attribute ID is carried in the Payload field:

1. UWB Session Identifier

11.7.2.9 Ranging Session Resume Response Message ID

This Ranging Session Resume Response Message ID is a response to Ranging Session Resume Request Message ID.

The following Attribute IDs are carried in the Payload field:

1. STS Index0
2. UWB Time0

11.7.3 Notification Protocol Type

11.7.3.1 Event Message ID

The Event Message ID is sent by the User Device and the Reader. The Payload field carries the Attribute IDs in Table 11-15.

Table 11-15 – Attribute IDs for Event Message ID

Attribute ID	Attribute ID Value
Busy	0
General Error	1
Reader Descriptor	2
RFU	3 - 255

11.7.3.1.1 Busy Attribute ID

The Attribute Length field is set to zero octet. The Attribute Value field is not present.

11.7.3.1.2 General Error Attribute ID

The Attribute Length field is set to one octet. The Attribute Value field carries the reason code enumerated in Table 11-16. Only Unknown Error reason code and Resource Unavailable SHALL be sent if the BleSK is not available for encryption. If the BleSK is available for encryption, any reason code is allowed to be sent.

Table 11-16 – Attribute Value for General Error Attribute ID

Reason Code	Value	Description
Unknown Error	0	No specific reason for failure is indicated.
Resource Unavailable	1	Indicates internal resource unavailability.
Wrong Parameters	2	Indicates use of unsupported message or message format.
URSK Unavailable	3	Indicates URSK corresponding to the UWB Session Identifier is not found.
RFU	4 - 255	RFU

11.7.3.1.3 Reader Descriptor Attribute ID

The Attribute Length field is set to variable octets. The Attribute Value field is enumerated in Table 11-17.

Table 11-17 – Attribute value for Reader Descriptor Attribute ID

Attribute Length	Attribute Value
Variable	The content of Reader Descriptor TLV object (including the tag 0xB5 and its length field), see Table 8-17.

11.7.3.2 Ranging Message ID

The Payload field carries the Attribute IDs in Table 11-18.

Table 11-18 – Attribute IDs for Ranging Message ID

Attribute ID	Attribute ID Value
Initiate Ranging Session	0
Initiate Ranging Session Resume	1
Initiate Ranging Session Setup Later	2
Initiate Ranging Session Resume Later	3
Secure Ranging Over UWB Radio Failed	4
Ranging Session Suspended	5
RFU	6 - 255

11.7.3.2.1 Initiate Ranging Session Attribute ID

The Attribute Length field is set to zero octet. The Attribute Value field is not present.

The User Device SHALL send this Attribute ID to trigger the Reader to initiate a new UWB ranging session. The Reader SHALL NOT send this Attribute ID.

11.7.3.2.2 Initiate Ranging Session Resume Attribute ID

The Attribute Length field is set to zero octet. The Attribute Value field is not present.

The User Device SHALL send this Attribute ID to trigger the Reader to initiate resumption of an existing UWB ranging session. The Reader SHALL NOT send this Attribute ID.

11.7.3.2.3 Initiate Ranging Session Setup Later Attribute ID

The Attribute Length field is set to zero octet. The Attribute Value field is not present.

The User Device SHALL send this Attribute ID after receiving Ranging Session Setup M1 Message ID from the Reader if the User Device cannot do UWB ranging session setup at this time.

The Reader SHALL send this Attribute ID after receiving Initiate Ranging Session Attribute ID from the User Device if the Reader cannot do UWB ranging session setup at this time.

11.7.3.2.4 Initiate Ranging Session Resume Later Attribute ID

The Attribute Length field is set to zero octet. The Attribute Value field is not present.

The User Device SHALL send this Attribute ID after receiving Ranging Session Resume Request Message ID from the Reader if the User Device cannot resume an existing UWB ranging session at this time.

The Reader SHALL send this Attribute ID after receiving Initiate Ranging Session Resume Attribute ID from the User Device if the Reader cannot resume an existing UWB ranging session at this time.

11.7.3.2.5 Secure Ranging Over UWB Radio Failed Attribute ID

The Attribute Length field is set to zero octet. The Attribute Value field is not present.

The Reader SHALL send this Attribute ID if it failed to establish secure ranging over UWB radio after receiving Ranging Session Setup M4 Message ID or Ranging Session Resume Response Message ID from the User Device.

11.7.3.2.6 Ranging Session Suspended Attribute ID

The Attribute Length field is set to zero octet. The Attribute Value field is not present.

Ranging Session Suspended Attribute ID provides a way for the User Device or the Reader to unilaterally suspend the current UWB ranging session. The User Device or the Reader SHALL send this Attribute ID if it is suspending the current UWB ranging session. The Reader or the User Device SHALL suspend the current UWB ranging session on receiving this Attribute ID.

11.7.3.3 Reader Status Changed Message ID

The Payload field SHALL carry State Attribute IDs defined in Table 11-19 and Table 11-20, respectively.

Table 11-19 – Attribute IDs for Reader Status Changed Message ID

Attribute ID	Attribute ID Value
State	0
RFU	1 - 255

11.7.3.3.1 State Attribute ID

The Attribute Length field is set to two octets. The bits [B15:B8] in the Attribute Value field indicate the operation source information enumerated in Table 11-20.

The bits [B7:B0] in the Attribute Value field indicate the Reader status information enumerated in the second byte field when the first byte field is equal to 0x01 in Table 8-18.

Table 11-20 – Operation source information in State Attribute ID

Operation Source Name	Attribute Value	Description
Unspecified	0	Operation that caused the state change came from unspecified source.
Manual	1	Operation that caused the state change came from manual behavior.
Auto	2	Operation that caused the state change came from the Reader automatically.
Schedule	3	Operation that caused the state change came from the Reader due to a schedule.
This User Device in Bluetooth LE + UWB Aliro Flow	4	Operation that caused the state change came in Bluetooth LE + UWB Aliro flow (see section 11.1.1) from the User Device that is the receiver of this Attribute ID in Reader Status Changed Message ID.
This User Device in NFC	5	Operation that caused the state change came in NFC from the User Device that is the receiver of this Attribute ID in Reader Status Changed Message ID.
This User Device in Bluetooth LE-Only Flow	6	Operation that caused the state change came in Bluetooth LE-Only Aliro flow (see section 11.1.2) from the User Device that is the receiver of this Attribute ID in Reader Status Changed Message ID.
Matter	7	Operation that caused the state change came from Matter [29].
RFU	8 - 255	RFU

11.7.3.4 Reader Status Access Protocol Completed Message ID

The Payload field SHALL carry the Reader Information Attribute ID. The Reader SHALL send Reader Status Access Protocol Completed Message ID after the Access Protocol is completed successfully with a User Device.

Table 11-21 – Attribute IDs for Reader Status Access Protocol Completed Message ID

Attribute ID	Attribute ID Value
Reader Information Attribute ID	0
RFU	1 - 255

11.7.3.4.1 Reader Information Attribute ID

The Attribute Length field is set to two octets. The bits [B15:B8] in the Attribute Value field indicate Reader capabilities enumerated in Table 11-22. The bits [B7:B0] in the Attribute Value field indicate the Reader status information enumerated in the second byte field when the first byte field is equal to 0x01 in Table 8-18.

A Reader SHALL support Unsolicited Reader Status Reporting with a value of either 1 or 2 when it supports, Bluetooth LE-Only Aliro flow (see section 11.1.2) or Bluetooth LE + UWB Aliro flow (see section 11.1.1).

Table 11-22 – Reader capability information in Reader Information Attribute ID

Reader capability name	Bit Position	Description
Unsolicited Reader Status Reporting	B15:B13	Value 0: RFU. Value 1: The Reader SHALL send Reader Status Changed Message ID to each connected User Device, to whom Reader has sent Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID, whenever the Reader state changes. Value 2: The Reader SHALL send Reader Status Changed Message ID to only this User Device, to whom Reader has sent Reader Status Access Protocol Completed Message ID carrying Reader Information Attribute ID, whenever this User Device causes the Reader state change.
RFU	B12:B8	RFU

11.7.3.5 RKE Request Message ID

The Payload field SHALL carry Action Attribute ID. The User Device sends RKE Request Message ID.

Table 11-23 – Attribute IDs for RKE Request Message ID

Attribute ID	Attribute ID Value
Action	0
RFU	1 - 255

11.7.3.6 Action Attribute ID

The Attribute Length field is set to one octet. The Attribute Value field is enumerated in Table 11-24.

Table 11-24 – Attribute Value for Action Attribute ID

Action Code	Attribute Value	Description
Secure	0	User Device SHALL send this action code to indicate to the Reader to go to secure state. Secure includes lock, arm, close etc.
Unsecure	1	User Device SHALL send this action code to indicate to the Reader to go to unsecure state. Unsecure includes unlock, disarm, open etc.
RFU	2 - 255	RFU

11.7.3.7 Initiate Access Protocol Message ID

The User Device SHALL send Initiate Access Protocol Message ID to trigger the Reader to initiate Access Protocol.

The Payload field carries Proprietary Information Attribute ID in Table 11-25.

Table 11-25 – Attribute ID in Initiate Access Protocol Message ID

Attribute ID	Attribute ID Value
Proprietary Information	0
RFU	1 - 255

11.7.3.7.1 Proprietary Information Attribute ID

The Attribute Length field is set to variable. The Attribute Value field is set to the content of the proprietary information TLV object (including the Tag 0xA5 byte and its length field) and optionally User Device Descriptor TLV object (including the Tag 0xB7 and its length field), as per SELECT response in section 10.2.1.2.

11.7.3.8 Initiate Access Protocol RKE Message ID

The User Device SHALL send Initiate Access Protocol RKE Message ID to trigger the Reader to initiate Access Protocol.

The Payload carries the Proprietary Information Attribute ID from Table 11-25. Other Attribute IDs are RFU.

11.7.4 Supplementary Service Protocol Type

11.7.4.1 Attribute ID

Attribute IDs for Supplementary Service Protocol Type are listed in Table 11-26.

Table 11-26 – Attribute IDs for Supplementary Service Protocol Type

Attribute ID	Attribute ID Value (decimal)
Device Event Count	0
UWB Device Time	1
UWB Device Time Uncertainty	2
UWB Clock Skew Measurement Available	3
Device Max PPM	4
Success	5
Retry Delay	6
Reserved	7-255

11.7.4.1.1 Device Event Count Attribute ID

The Attribute Length field is set to eight octets.

The Attribute Value field has device event count value as described in DeviceEventCount parameter in Time_Sync in [2].

11.7.4.1.2 UWB Device Time Attribute ID

The Attribute Length field is set to eight octets.

The Attribute Value field has UWB device time value as described in UWB_Device_Time parameter in Time_Sync in [2].

11.7.4.1.3 UWB Device Time Uncertainty Attribute ID

The Attribute Length field is set to one octet.

The Attribute Value field has UWB device time uncertainty value as described in UWB_Device_Time_Uncertainty parameter in Time_Sync in [2].

11.7.4.1.4 UWB Clock Skew Measurement Available Attribute ID

The Attribute Length field is set to one octet.

The Attribute Value field has UWB clock skew measurement availability indication as described in UWB_Clock_Skew_Measurement_available parameter in Time_Sync in [2].

11.7.4.1.5 Device Max PPM Attribute ID

The Attribute Length field is set to two octets.

The Attribute Value field has worst case clock skew of device UWB clock as described in Device_max_PPM in Time_Sync in [2].

11.7.4.1.6 Success Attribute ID

The Attribute Length field is set to one octet.

The Attribute Value field has indication of status of Bluetooth LE Timesync procedure as described in Success parameter in Time_Sync in [2].

11.7.4.1.7 Retry Delay Attribute ID

The Attribute Length field is set to two octets.

The Attribute Value field has minimum delay value required by the User Device until the Reader SHOULD trigger a new Bluetooth LE Timesync, as described in RetryDelay parameter in Time_Sync in [2].

11.7.4.2 Time Sync Message ID

Time Sync Message ID is used by the User Device to provide Bluetooth LE Timesync payload to the Reader.

The following Attribute IDS are carried in the Payload field:

1. Device Event Count
2. UWB Device Time
3. UWB Device Time Uncertainty
4. UWB Clock Skew Measurement Available
5. Device Max PPM
6. Success
7. Retry Delay

11.7.5 3rd Party App Protocol Type

11.7.5.1 Pass Through Message ID

This 3rd Party App Message ID is used by the Reader OEM's app and the Reader to exchange data with each other over the same L2CAP channel used in the Aliro service. Support for this type is OPTIONAL for the User Device and the Reader.

The Length field is set to 3 + the length of the Payload field.

The Length field is followed by an IEEE OUI or CID of 3 octets. The OUI or CID field is set to variable value defined in IEEE Registration Authority. The Payload field carries the proprietary payload known to the Reader OEM's app and Reader. Length of the payload is variable and depends on the User Device/Reader payload size over Bluetooth LE limitations.

11.8 Aliro Message Security

The Aliro messages with Protocol Type "AP" SHALL have their payloads encrypted and authenticated as per section 8.

The Aliro messages with Protocol Type "UWB Ranging Service", "Notification", "Supplementary Service", and "3rd Party App" SHALL have their payloads encrypted and authenticated as per sections 11.8.1 and 11.8.2.

11.8.1 Session Key Derivation

The 32 bytes key BleSKReader SHALL be derived as per section 8.3.1.5 using BleSK (computed as per as per section 8.3.1.12 or section 8.3.1.13) as input_key_material, "BleSKReader" (utf8-encoded string) as info, 32 as key_material_length, salt as below.

The 32 bytes key BleSKDevice SHALL be derived as per section 8.3.1.5 using BleSK (computed as per section 8.3.1.12 or section 8.3.1.13) as input_key_material, "BleSKDevice" (utf8-encoded string) as info, 32 as key_material_length, salt as below.

The "salt" input of the key derivation contains: the list of protocol versions supported by the Reader referenced as "Supported ALIRO BLE UWB Protocol Version" in Table 11-7 denoted as reader_supported_versions below, the protocol version selected by the User Device referenced as "Selected ALIRO BLE UWB Protocol Version" in Table 11-6 denoted as user_device_selected_version below. The "salt" is generated as follow:

salt = reader_supported_versions || user_device_selected_version

11.8.2 Encryption and Authentication

The Reader SHALL encrypt using the session key BleSKReader, the User Device SHALL encrypt using the session key BleSKDevice.

The Reader and the User Device SHALL keep a separate message counter for each session key (ble_device_counter is associated to BleSKDevice and ble_reader_counter is associated to BleSkReader). The Reader and User Device SHALL initialize a session-bound `ble_device_counter` to value 0x00000001 and session-bound `ble_reader_counter` to value 0x00000001. A message counter SHALL NOT be reused in any future encryption using the same key.

The Payload field of the Aliro message SHALL contain the encrypted_payload and authentication_tag concatenated in that order. The Length field SHALL be adjust accordingly to take into account the authentication_tag.

The Payload field of the Aliro messages transmitted from the User Device to the Reader SHALL be encrypted and `encrypted_payload` and `authentication_tag` generated according section 8.3.1.6 using the session-bound key BleSKDevice as SKDevice, the Aliro message unencrypted Payload field as payload, ble_device_counter as device_counter, the 4 bytes formed by Protocol Header || Message ID || Length of plain payload as aad. The device_counter output from the procedure is used to update `expedited_device_counter`.

The Payload field of the Aliro messages received by the Reader from the User Device SHALL be decrypted and have their authenticity verified according to section 8.3.1.7 using the Aliro message Payload field (minus the last 16 bytes) as encrypted_payload, the last 16 bytes of Aliro message Payload field as authentication_tag, BleSKDevice as SKDevice, ble_device_counter as device_counter, the 4 bytes formed by Protocol Header || Message ID || Length of plain payload as aad. Only if the resulting authentication_tag_verified boolean value indicates no failure occurred, the decrypted payload content can be processed by the Reader. The device_counter output from the procedure is used to update `expedited_device_counter`.

The Payload field of the Aliro messages transmitted from the Reader to the User Device SHALL be encrypted and `encrypted_payload` and `authentication_tag` generated according section 8.3.1.8 using the session-bound key BleSKReader as SKReader, the Aliro message unencrypted Payload field as payload, ble_reader_counter as reader_counter, the 4 bytes formed by Protocol Header || Message ID || Length of plain payload as aad. The reader_counter output from the procedure is used to update `expedited_reader_counter`.

The Payload field of the Aliro messages received by the User Device from the Reader SHALL be decrypted and have their authenticity verified according to section 8.3.1.9 using the Aliro message Payload field (minus the last 16 bytes) as encrypted_payload, the last 16 bytes of Aliro message Payload field as authentication_tag, BleSKReader as SKReader, ble_reader_counter as reader_counter, the 4 bytes formed by Protocol Header || Message ID || Length of plain payload as aad. Only if the resulting authentication_tag_verified boolean value indicates no failure occurred, the decrypted payload content can be processed by the User Device. The reader_counter output from the procedure is used to update `expedited_reader_counter`.

The transaction SHALL be aborted and the Bluetooth LE connection with the peer terminated in case an invalid authentication_tag is received, or a message counter value reaches 0xFFFF.

11.9 Aliro Message Rules

The transmitter (Reader or User Device) of a Message ID can get the corresponding Message ID reply from the receiver with non-predictable delay depending upon the receiver's resource availability. It is also desirable to detect when the receiver becomes unresponsive to take appropriate actions such as releasing computation resources at the transmitter of the Message ID. It should be noted that Bluetooth LE with L2CAP is a reliable transport between the transmitter and the receiver, therefore transmission loss detection and recovery in Bluetooth LE is out of scope of this specification, consequently the following rules apply as long the L2CAP connection remains established between the User Device and the Reader.

Table 11-27 enumerates the various Message IDs and their corresponding permitted Message ID reply. Table 11-28 enumerates Message IDs that do not have a corresponding Message ID reply.

Table 11-27 – Aliro Message IDs that have a responseTimeout rule

Message ID	Message ID Reply
Initiate Access Protocol	Any of the following: 1. AP_RQ carrying AUTH0 command 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID
Initiate Access Protocol RKE	Any of the following: 1. AP_RQ carrying AUTH0 command 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID
AP_RQ carrying AUTH0 command	Any of the following: 1. AP_RS carrying AUTH0 response 2. Event carrying Busy Attribute ID
AP_RS carrying AUTH0 response	Any of the following: 1. AP_RQ carrying LOAD CERT command 2. AP_RQ carrying AUTH1 command 3. AP_RQ carrying EXCHANGE command 4. Event carrying General Error Attribute ID 5. Event carrying Busy Attribute ID 6. AP_RQ carrying AUTH0 command, if AUTH0 command chaining occurred 7. AP_RQ carrying GET_RESPONSE command, if AUTH0 response chaining occurred
AP_RQ carrying LOAD CERT command	Any of the following: 1. AP_RS carrying LOAD CERT response 2. Event carrying Busy Attribute ID
AP_RS carrying LOAD CERT response	Any of the following: 1. AP_RQ carrying AUTH1 command 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID 4. AP_RQ carrying LOAD CERT command, if LOAD CERT command chaining occurred
AP_RQ carrying AUTH1 command	Any of the following: 1. AP_RS carrying AUTH1 command response

Message ID	Message ID Reply
	2. Event carrying Busy Attribute ID
AP_RS carrying AUTH1 response	Any of the following: 1. AP_RQ carrying EXCHANGE command 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID 4. AP_RQ carrying AUTH1 command, if AUTH1 command chaining occurred 5. AP_RQ carrying GET_RESPONSE command, if AUTH1 response chaining occurred
AP_RQ carrying EXCHANGE command	Any of the following: 1. AP_RS carrying EXCHANGE response 2. Event carrying Busy Attribute ID
AP_RS carrying EXCHANGE response	Any of the following: 1. AP_RQ carrying ENVELOPE command 2. AP_RQ carrying EXCHANGE command 3. Reader Status Access Protocol Completed carrying Reader Information Attribute ID 4. Event carrying General Error Attribute ID 5. Event carrying Busy Attribute ID 6. AP_RQ carrying GET_RESPONSE command, if EXCHANGE response chaining occurred
AP_RQ carrying ENVELOPE command	Any of the following: 1. AP_RS carrying ENVELOPE response command 2. Event carrying Busy Attribute ID
AP_RS carrying ENVELOPE response command	Any of the following: 1. Reader Status Access Protocol Completed carrying Reader Information Attribute ID 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID 4. AP_RQ carrying ENVELOPE command 5. AP_RQ carrying GET_RESPONSE command, if ENVELOPE response chaining occurred 6. AP_RQ carrying EXCHANGE command

Message ID	Message ID Reply
AP_RQ carrying GET_RESPONSE command	Any of the following: 1. AP_RS carrying GET_RESPONSE response command 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID
AP_RS carrying GET_RESPONSE response command	Any of the following: 1. Reader Status Access Protocol Completed carrying Reader Information Attribute ID 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID 4. AP_RQ carrying GET_RESPONSE command, if response chaining occurred 5. AP_RQ carrying EXCHANGE command, if Envelope response chaining occurred, or AUTH1 response chaining occurred, or EXCHANGE response chaining occurred 6. AP_RQ carrying LOAD CERT command, if AUTH0 response chaining occurred 7. AP_RQ carrying AUTH1 command, if AUTH0 response chaining occurred
Ranging carrying Initiate Ranging Session Attribute ID	Any of the following: 1. Ranging Session Setup M1 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID 4. Ranging carrying Initiate Ranging Session Setup Later Attribute ID
Ranging Session Setup M1	Any of the following: 1. Ranging Session Setup M2 2. Ranging carrying Initiate Ranging Session Setup Later Attribute ID 3. Event carrying General Error Attribute ID 4. Event carrying Busy Attribute ID
Ranging Session Setup M2	Any of the following: 1. Ranging Session Setup M3 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID

Message ID	Message ID Reply
Ranging Session Setup M3	Any of the following: 1. Ranging Session Setup M4 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID
Ranging Session Suspend Request	Any of the following: 1. Ranging Session Suspend Response 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID
Ranging carrying Initiate Ranging Session Resume Attribute ID	Any of the following: 1. Ranging Session Resume Request 2. Event carrying General Error Attribute ID 3. Event carrying Busy Attribute ID 4. Ranging carrying Initiate Ranging Session Resume Later Attribute ID
Ranging Session Resume Request	Any of the following: 1. Ranging Session Resume Response 2. Ranging carrying Initiate Ranging Session Resume Later Attribute ID 3. Event carrying General Error Attribute ID 4. Event carrying Busy Attribute ID

Table 11-28 – Aliro Message IDs that do not have responseTimeout rule

Message ID
Ranging carrying Initiate Ranging Session Setup Later Attribute ID
Ranging carrying Initiate Ranging Session Resume Later Attribute ID
Ranging carrying Secure Ranging Over UWB Radio Failed Attribute ID
Ranging carrying Ranging Session Suspended Attributed ID
RKE Request
Reader Status Access Protocol Completed
Reader Status Changed

Message ID
Time Sync
Pass Through
Ranging Session Setup M4
Ranging Session Resume Response
Ranging Session Suspend Response

11.9.1 Receiver Side Rules

On receiving any Message ID in first column of Table 11-27, the receiver SHALL set for this Message ID, response timer equal to responseTimeout, and start its countdown. On receiving any Message ID in Table 11-28, the receiver SHALL NOT set response timer. The responseTimeout is set to 1500 ms in this specification. Figure 11-4 is informative depiction of receiver side Aliro message rules.

While the response timer is not expired at the receiver:

1. The receiver SHALL send the Message ID reply in the second column of Table 11-27 in response to the corresponding received Message ID in the first column in Table 11-27. The receiver SHALL delete the response timer after transmitting Message ID reply other than Event Message ID carrying Busy Attribute ID in response to the corresponding received Message ID in the first column of Table 11-27.
2. The receiver sends Event Message ID carrying Busy Attribute ID in response to the corresponding received Message ID in the first column of Table 11-27, if the receiver needs more time beyond the response timer for processing. The receiver SHALL reset the response timer to responseTimeout after transmitting Event Message ID carrying Busy Attribute ID in response to the corresponding received Message ID in the first column of Table 11-27. The number of times a receiver can send Event Message ID carrying Busy Attribute ID in response to a Message ID is implementation specific and outside scope of this specification.

The receiver can incur an exception such that response timer is expired, and the receiver is unable to transmit the Message ID reply in the second column of Table 11-27 in response to the corresponding received Message ID in the first column of Table 11-27. In this case, the transmitter of the Message ID SHALL initiate Bluetooth LE connection teardown according to section 11.6.

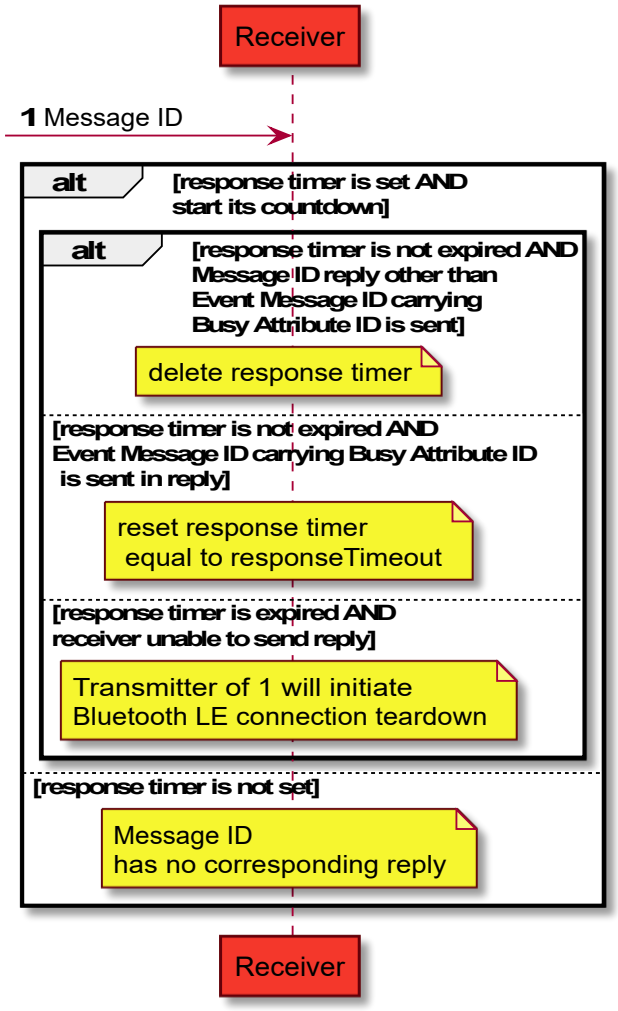


Figure 11-4 – Receiver side Aliro message rules

11.9.2 Transmitter Side Rules

On sending any Message ID in the first column of Table 11-27, the transmitter SHALL set for this Message ID, response timer equal to responseTimeout, and start its countdown. On sending any Message ID in Table 11-28, the transmitter SHALL NOT set response timer. The responseTimeout is set to 1500 ms in this specification. Figure 11-5 is informative depiction of transmitter side Aliro message rules.

While the response timer is not expired at the transmitter:

1. The transmitter SHALL delete the response timer after receiving the Message ID reply other than the Event Message ID carrying Busy Attribute ID in response to the corresponding transmitted Message ID in the first column of Table 11-27.
2. The transmitter SHALL initiate Bluetooth LE connection teardown after receiving the Event Message ID carrying General Error Attribute ID in response to the corresponding transmitted Message ID in the first column of Table 11-27.
3. The transmitter SHALL reset the response timer to responseTimeout after receiving Event Message ID carrying Busy Attribute ID in response to the corresponding transmitted Message ID

in the first column in Table 11-27. To avoid infinitely waiting for a Message ID reply other than Event Message ID carrying Busy Attribute ID, in response to the corresponding transmitted Message ID in the first column in Table 11-27, the transmitter can initiate Bluetooth LE connection teardown while the response timer is not expired. This behavior is implementation specific and outside scope of this specification.

If the response timer is expired, the transmitter SHALL initiate Bluetooth LE connection teardown.

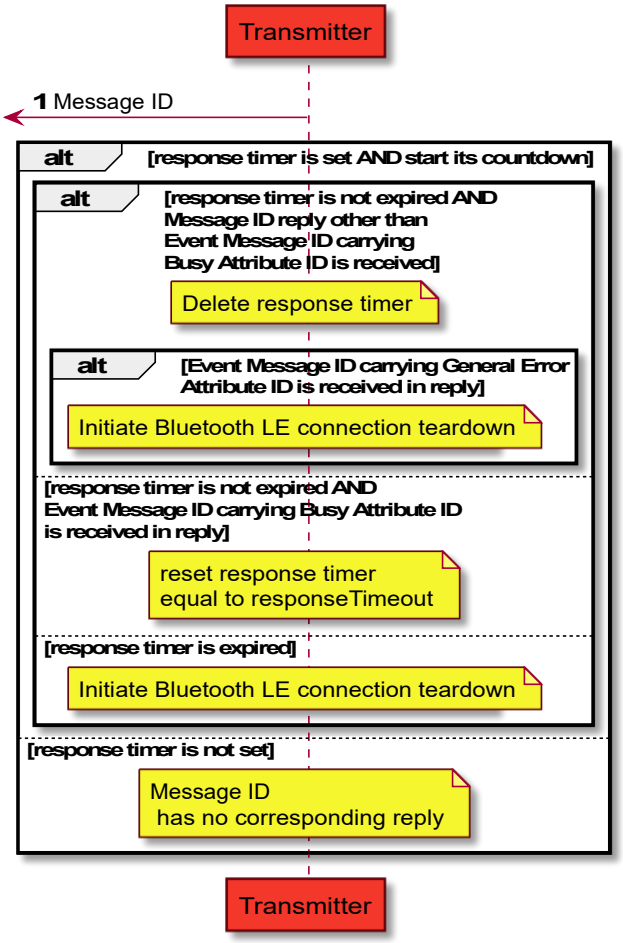


Figure 11-5 – Transmitter side Aliro message rules

11.9.3 Aliro Message Race Condition Rules

‘Sending’ in section 11.9.2 means submitting an Aliro message from the Aliro component to the system software on the User Device or the Reader for transmission over Bluetooth LE to the Reader or the User Device.

‘Receiving’ in section 11.9.1 means receipt of an Aliro message at the Aliro component from the system software on the User Device or the Reader after reception over Bluetooth LE from the Reader or the User Device.

There is a delay between submitting an Aliro message to the system software and its transmission from the Bluetooth LE interface. Likewise, there is a delay between receipt of an Aliro message at the Bluetooth LE interface and its reception at the Aliro component.

These delays at the Reader and the User Device yield out-of-order Aliro message exchanges that do not conform to Table 11-27. The out-of-order Aliro message exchanges scenarios are listed in Table 11-29 and its resolutions are described in this section.

Table 11-29 Out-of-order Aliro message exchanges scenarios

Out-of-Order Index	Reader Sends	User Device Sends
1	Ranging Session Setup M1 Message ID	Ranging Message ID carrying Initiate Ranging Session Attribute ID
2		Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID
3		Ranging Session Suspend Request Message ID
4		Ranging Message ID carrying Ranging Session Suspended Attributed ID
5	Ranging Session Resume Request Message ID	Ranging Message ID carrying Initiate Ranging Session Attribute ID
6		Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID
7		Ranging Session Suspend Request Message ID
8		Ranging Message ID carrying Ranging Session Suspended Attribute ID
9	Ranging Session Suspend Request Message ID	Ranging Message ID carrying Initiate Ranging Session Attribute ID
10		Ranging Message ID carrying Ranging Session Resume Attribute ID
11		Ranging Session Suspend Request Message ID

Out-of-Order Index	Reader Sends	User Device Sends
12		Ranging Message ID carrying Ranging Session Suspended Attribute ID
13	Ranging Message ID carrying Ranging Session Suspended Attribute ID	Ranging Message ID carrying Initiate Ranging Session Attribute ID
14		Ranging Message ID carrying Ranging Session Resume Attribute ID
15		Ranging Session Suspend Request Message ID
16		Ranging Message ID carrying Ranging Session Suspended Attribute ID
17	Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attribute ID	Ranging Message ID carrying Initiate Ranging Session Attribute ID
18		Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID
19		Ranging Session Suspend Request Message ID
20		Ranging Message ID carrying Ranging Session Suspended Attribute ID

11.9.3.1 Out-of-order Index 1

Figure 11-6 illustrates the out-of-order index 1 scenario in Table 11-29.

Reader sends Ranging Session Setup M1 Message ID and receives Ranging Message ID carrying Initiate Ranging Session Attribute ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Message ID carrying Initiate Ranging Session Attribute ID and receives Ranging Session Setup M1 Message ID that is an expected reply according to Table 11-27.

In this case, the Reader SHALL ignore and discard the Ranging Message ID carrying Initiate Ranging Session Attribute ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Session Setup M1 Message ID it sent.

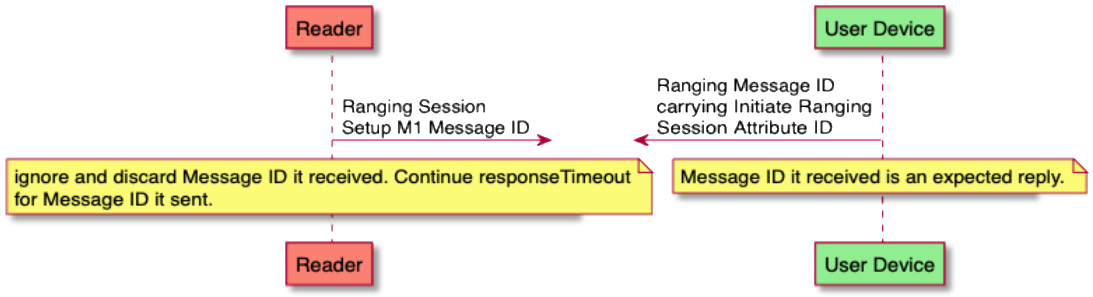


Figure 11-6 Aliro message exchange out-of-order index 1

11.9.3.2 Out-of-order Index 2

This combination does not occur because of rules in .

11.9.3.3 Out-of-order Index 3

11.9.3.4 Out-of-order Index 4

11.9.3.5 Out-of-order Index 5

Figure 11-7 illustrates the out-of-order index 5 scenario in Table 11-29.

Reader sends Ranging Session Resume Request Message ID and receives Ranging Message ID carrying Initiate Ranging Session Attribute ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Message ID carrying Initiate Ranging Session Attribute ID and receives Ranging Session Resume Request Message ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL discard Ranging Session Resume Request Message ID it sent and delete the corresponding responseTimeout set according to section 11.9.2. The Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Message ID carrying Initiate Ranging Session Attribute ID it received.

The User Device SHALL ignore and discard Ranging Session Resume Request Message ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Message ID carrying Initiate Ranging Session Attribute ID it sent.

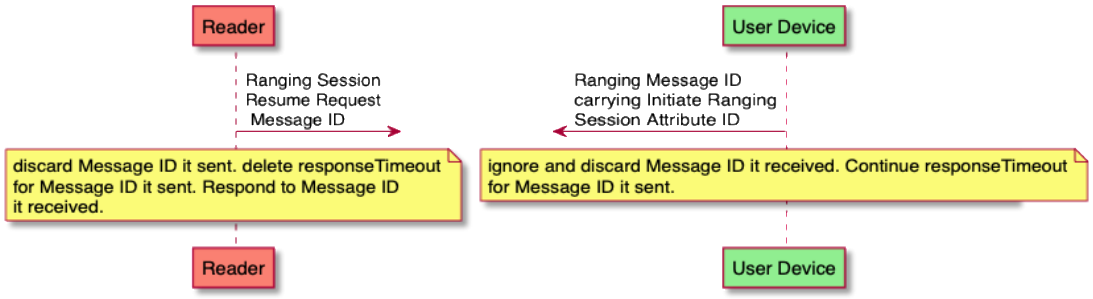


Figure 11-7 Aliro message exchange out-of-order index 5

11.9.3.6 Out-of-order Index 6

Figure 11-8 illustrates the out-of-order index 6 scenario in Table 11-29.

Reader sends Ranging Session Resume Request Message ID and receives Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Session Suspend Request Message ID and receives Ranging Session Resume Request Message ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL discard Ranging Session Resume Request Message ID it sent and delete the corresponding responseTimeout set according to section 11.9.2. The Reader SHALL follow section 11.9.1 and to Table 11-27 in responding to Ranging Session Suspend Request Message ID it received.

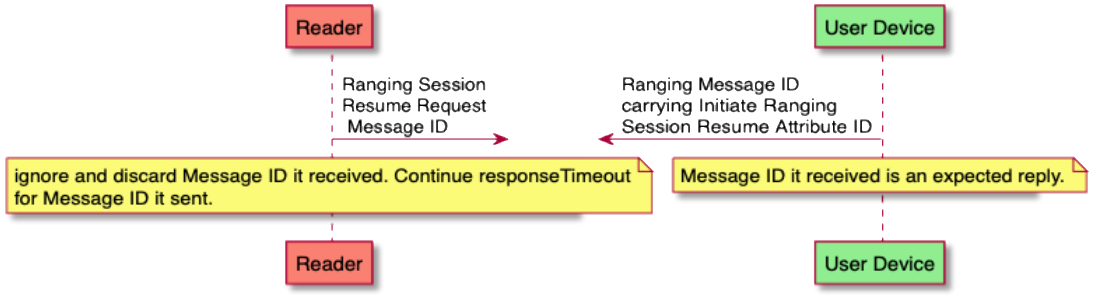


Figure 11-8 Aliro message exchange out-of-order index 6

11.9.3.7 Out-of-order Index 7

Figure 11-9 illustrates the out-of-order index 7 scenario in Table 11-29.

Reader sends Ranging Session Resume Request Message ID and receives Ranging Session Suspend Request Message ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Session Suspend Request Message ID and receives Ranging Session Resume Request Message ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL discard Ranging Session Resume Request Message ID it sent and delete the corresponding responseTimeout set according to section 11.9.2. The Reader SHALL follow section 11.9.1 and to Table 11-27 in responding to Ranging Session Suspend Request Message ID it received.

The User Device SHALL ignore and discard Ranging Session Resume Request Message ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Session Suspend Request Message ID it sent.

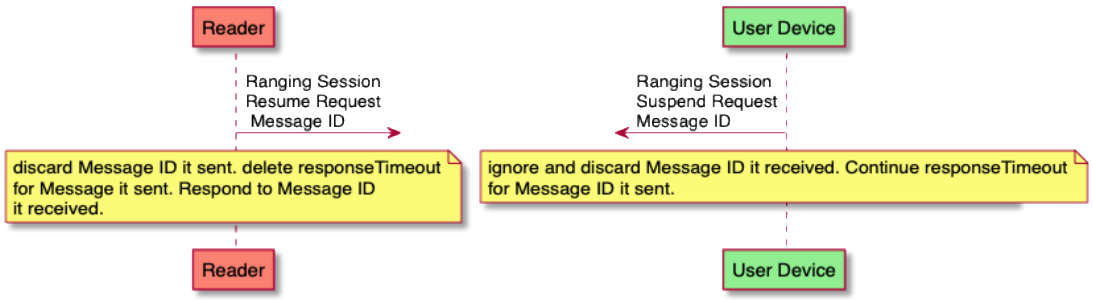


Figure 11-9 Aliro message exchange out-of-order index 7

11.9.3.8 Out-of-order Index 8

Figure 11-10 illustrates the out-of-order index 8 scenario in Table 11-29.

Reader sends Ranging Session Resume Request Message ID and receives Ranging Message ID carrying Ranging Session Suspended Attribute ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Message ID carrying Ranging Session Suspended Attributed ID and this Message ID has no corresponding response in Table 11-28.

In this case, the Reader SHALL ignore and discard Ranging Message ID carrying Ranging Session Suspended Attribute ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Session Resume Request Message ID it sent.

The User Device SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Session Resume Request Message ID it received.

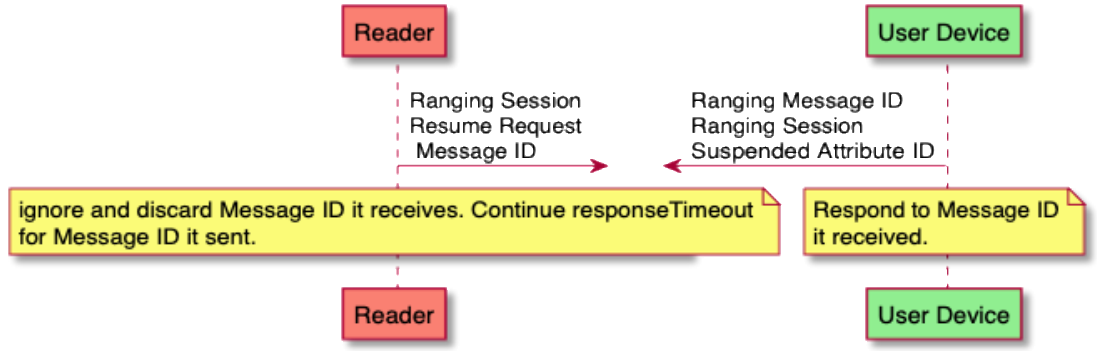


Figure 11-10 Aliro message exchange out-of-order index 8

11.9.3.9 Out-of-order Index 9

Figure 11-11 illustrates the out-of-order index 9 scenario in Table 11-29.

Reader sends Ranging Session Suspend Request Message ID and receives Ranging Message ID carrying Initiate Ranging Session Attribute ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Message ID carrying Initiate Ranging Session Attribute ID and receives Ranging Session Suspend Request Message ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL discard Ranging Session Suspend Request Message ID it sent and delete the corresponding responseTimeout set according to section 11.9.2. The Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Message ID carrying Initiate Ranging Session Attribute ID it received.

The User Device SHALL ignore and discard Ranging Session Suspend Request Message ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Message ID carrying Initiate Ranging Session Attribute ID it sent.

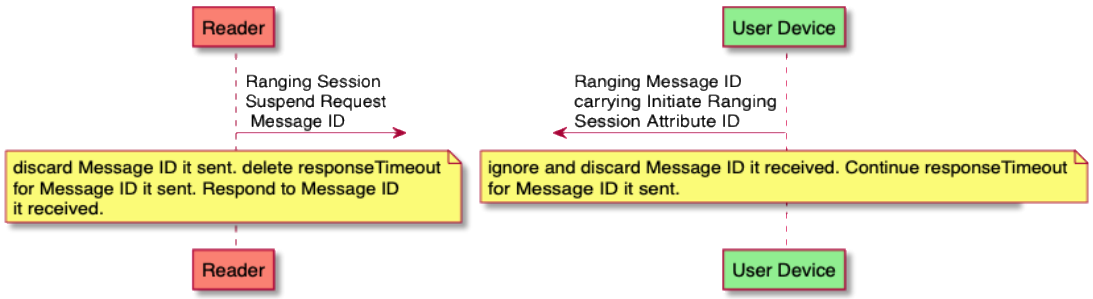


Figure 11-11 Aliro message exchange out-of-order index 9

11.9.3.10 Out-of-order Index 10

Figure 11-12 illustrates the out-of-order index 10 scenario in Table 11-29.

Reader sends Ranging Session Suspend Request Message ID and receives Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID and receives Ranging Session Suspend Request Message ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL discard Ranging Session Suspend Request Message ID it sent and delete the corresponding responseTimeout set according to section 11.9.2. The Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID it received.

The User Device SHALL ignore and discard Ranging Session Suspend Request Message ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID it sent.

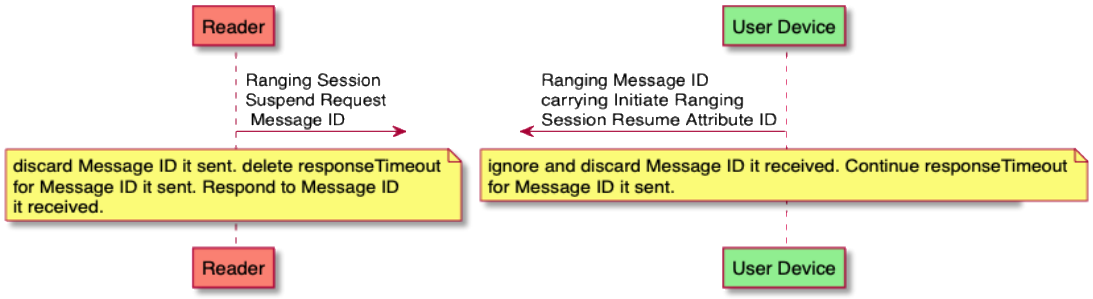


Figure 11-12 Aliro message exchange out-of-order index 10

11.9.3.11 Out-of-order Index 11

Figure 11-13 illustrates the out-of-order index 11 scenario in Table 11-29.

Reader sends Ranging Session Suspend Request Message ID and receives Ranging Message Session Suspend Request ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Session Suspend Request Message ID and receives Ranging Session Suspend Request Message ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL discard Ranging Session Suspend Request Message ID it sent and delete the corresponding responseTimeout set according to section 11.9.2. The Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Session Suspended Message ID it received.

The User Device SHALL ignore and discard Ranging Session Suspend Request Message ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Session Suspend Request Message ID it sent.

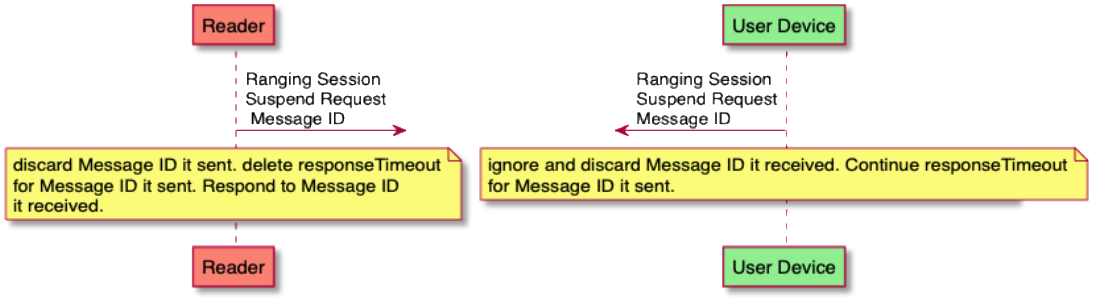


Figure 11-13 Aliro message exchange out-of-order index 11

11.9.3.12 Out-of-order Index 12

Figure 11-14 illustrates the out-of-order index 12 scenario in Table 11-29.

Reader sends Ranging Session Suspend Request Message ID and receives Ranging Message ID carrying Ranging Session Suspended Attribute ID that is not an expected reply according to Table 11-27.

User Device sends Ranging Message ID carrying Ranging Session Suspended Attributed ID and this Message ID has no corresponding response in Table 11-28.

In this case, the Reader SHALL ignore and discard Ranging Message ID carrying Ranging Session Suspended Attribute ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Session Suspend Request Message ID it sent.

The User Device SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Session Suspend Request Message ID it received.

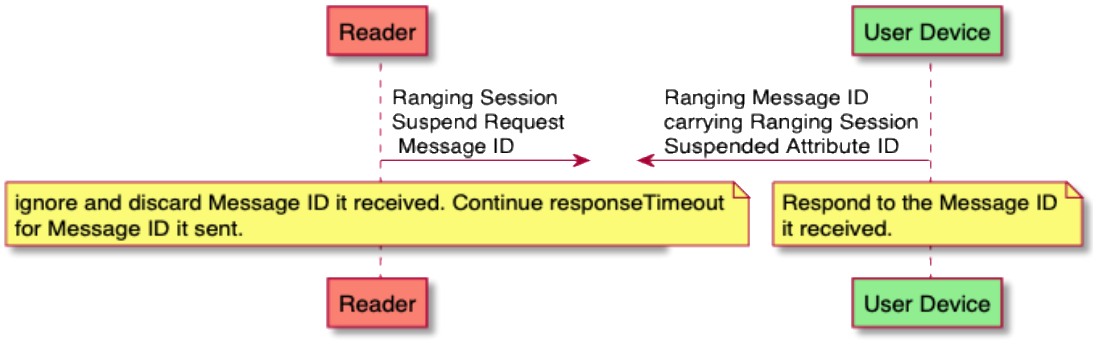


Figure 11-14 Aliro message exchange out-of-order index 12

11.9.3.13 Out-of-order Index 13

Figure 11-15 illustrates the out-of-order index 13 scenario in Table 11-29.

Reader sends Ranging Message ID carrying Ranging Session Suspended Attributed ID and this Message ID has no corresponding response in Table 11-28.

User Device sends Ranging Message ID carrying Initiate Ranging Session Attribute ID and receives Ranging Message ID carrying Ranging Session Suspended Attribute ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Message ID carrying Initiate Ranging Session Attribute ID it received.

The User Device SHALL ignore and discard Ranging Message ID carrying Ranging Session Suspended Attribute ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Message ID carrying Initiate Ranging Session Attribute ID it sent.

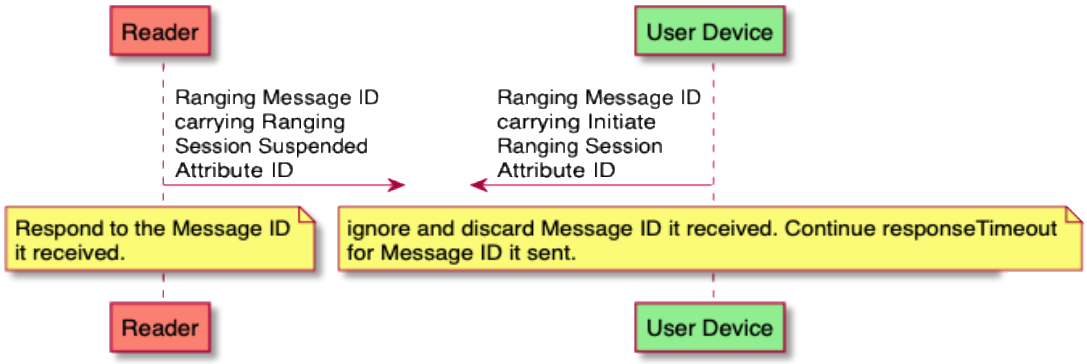


Figure 11-15 Aliro message exchange out-of-order index 13

11.9.3.14 Out-of-order Index 14

Figure 11-16 illustrates the out-of-order index 14 scenario in Table 11-29.

Reader sends Ranging Message ID carrying Ranging Session Suspended Attributed ID and this Message ID has no corresponding response in Table 11-28.

User Device sends Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID and receives Ranging Message ID carrying Ranging Session Suspended Attribute ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID it received.

The User Device SHALL ignore and discard Ranging Message ID carrying Ranging Session Suspended Attribute ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID it sent.

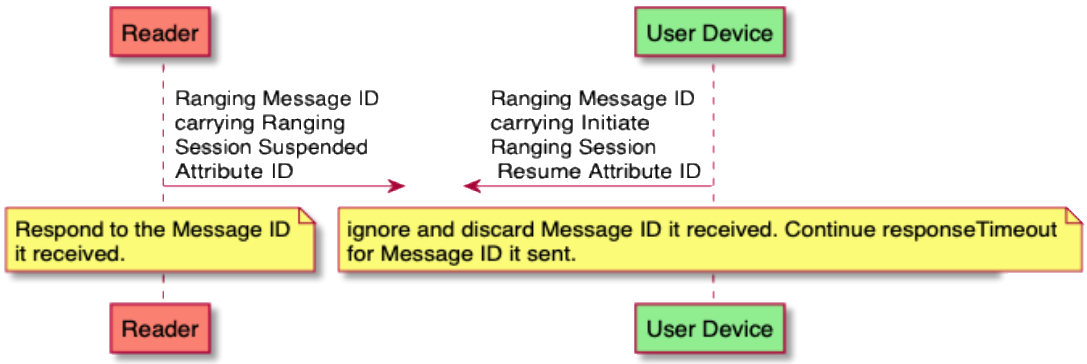


Figure 11-16 Aliro message exchange out-of-order index 14

11.9.3.15 Out-of-order Index 15

Figure 11-17 illustrates the out-of-order index 15 scenario in Table 11-29.

Reader sends Ranging Message ID carrying Ranging Session Suspended Attributed ID and this Message ID has no corresponding response in Table 11-28.

User Device sends Ranging Session Suspend Request Message ID and receives Ranging Message ID carrying Ranging Session Suspended Attribute ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Session Suspend Request Message ID it received.

The User Device SHALL ignore and discard Ranging Message ID carrying Ranging Session Suspended Attribute ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Session Suspend Request Message ID it sent.

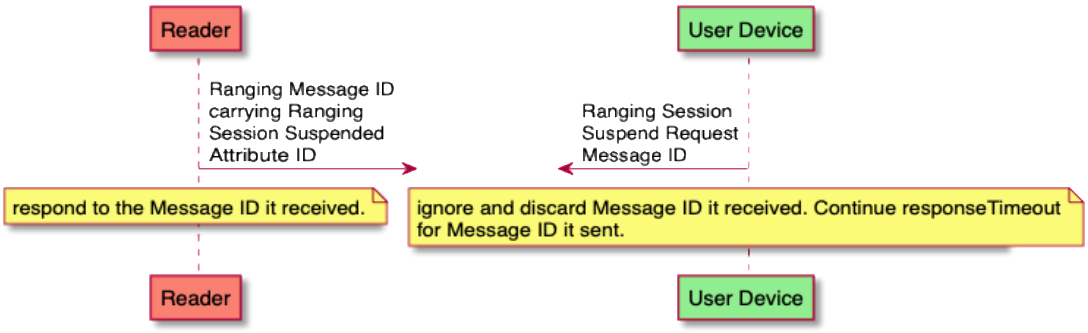


Figure 11-17 Aliro message exchange out-of-order index 15

11.9.3.16 Out-of-order Index 16

Reader sends Ranging Message ID carrying Ranging Session Suspended Attributed ID and this Message ID has no corresponding response in Table 11-28.

User Device sends Ranging Message ID carrying Ranging Session Suspended Attributed ID and this Message ID has no corresponding response in Table 11-28.

11.9.3.17 Out-of-order Index 17

Figure 11-18 illustrates the out-of-order index 17 scenario in Table 11-29.

Reader sends Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attributed ID and this Message ID has no corresponding response in Table 11-28.

User Device sends Ranging Message ID carrying Initiate Ranging Session Attribute ID and receives Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attributed ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Message ID carrying Initiate Ranging Session Attribute ID it received.

The User Device SHALL ignore and discard Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attributed ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Message ID carrying Initiate Ranging Session Attribute ID it sent.

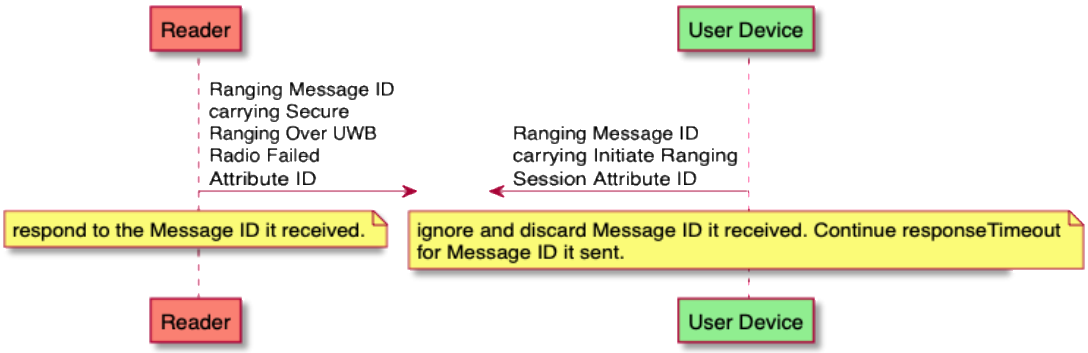


Figure 11-18 Aliro message exchange out-of-order index 17

11.9.3.18 Out-of-order Index 18

Figure 11-19 illustrates the out-of-order index 18 scenario in Table 11-29.

Reader sends Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attributed ID and this Message ID has no corresponding response in Table 11-28.

User Device sends Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID and receives Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attributed ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL follow section 11.9.1 and Table 11-27 in responding to Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID it received.

The User Device SHALL ignore and discard Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attributed ID it receives and continues its responseTimeout countdown according to section 11.9.2 for Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID it sent.

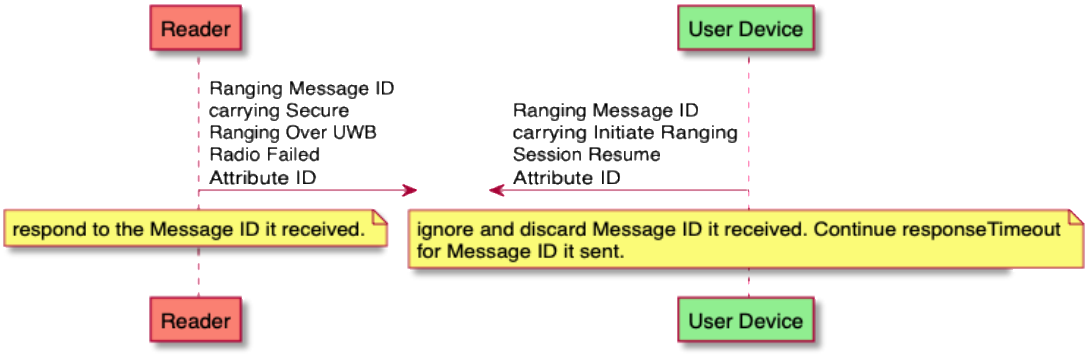


Figure 11-19 Aliro message exchange out-of-order index 18

11.9.3.19 Out-of-order Index 19

Figure 11-20 illustrates the out-of-order index 19 scenario in Table 11-29.

Reader sends Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attributed ID and this Message ID has no corresponding response in Table 11-28.

User Device sends Ranging Session Suspend Request Message ID and receives Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attributed ID that is not an expected reply according to Table 11-27.

In this case, the Reader SHALL ignore and discard Ranging Session Suspend Request Message ID it received.

The User Device SHALL discard Ranging Session Suspend Request Message ID it sent and delete its corresponding responseTimeout countdown set according to section 11.9.2 for Ranging Session Suspend Request Message ID it sent.

User Device behavior on receiving Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attribute ID is described in section 11.1.1.3.

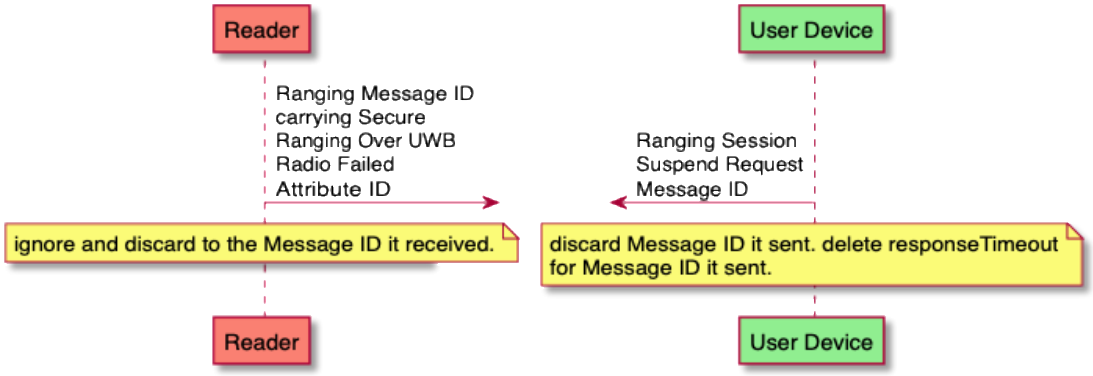


Figure 11-20 Aliro message exchange out-of-order index 19

11.9.3.20 Out-of-order Index 20

Reader sends Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attribute ID and this Message ID has no corresponding response in Table 11-28.

User Device sends Ranging Message ID carrying Ranging Session Suspended Attributed ID and this Message ID has no corresponding response in Table 11-28.

User Device behavior on receiving Ranging Message ID carrying Secure Ranging Over UWB Radio Failed Attribute ID is described in section 11.1.1.3.

11.9.4 Other Aliro Message Rules

The rules described in this section limit the out-of-order Aliro message combinations that can occur.

11.9.4.1 UWB Ranging Session Setup

1. Ranging Session Setup M1 Message ID SHALL NOT be sent without receiving a corresponding Ranging Message ID carrying Initiate Ranging Session Attribute ID, after receiving Ranging Session Setup M4 Message ID in this Bluetooth LE connection.
2. Ranging Message ID carrying Initiate Ranging Session Attribute ID SHALL NOT be sent in the presence of an active UWB ranging session.

3. Ranging Session Setup M1 Message ID SHALL NOT be sent in the presence of an active UWB ranging session.
4. Ranging Session Setup M2 Message ID SHALL NOT be sent without receiving the corresponding Ranging Session Setup M1 Message ID.
5. Ranging Session Setup M3 Message ID SHALL NOT be sent without receiving the corresponding Ranging Session Setup M2 Message ID.
6. Ranging Session Setup M4 Message ID SHALL NOT be sent without receiving the corresponding Ranging Session Setup M3 Message ID.
7. An active UWB ranging session SHALL be suspended, if it exists, before sending Ranging Message ID carrying Initiate Ranging Session Attribute ID.

Ranging Message ID carrying Initiate Ranging Session Setup Later Attribute ID SHALL NOT be sent without receiving the corresponding Ranging Session Setup M1 Message ID.

11.9.4.2 Suspend and Resume UWB Ranging Session

1. Ranging Session Suspend Request Message ID and Ranging Message ID carrying Ranging Session Suspended Attribute ID SHALL NOT be sent without presence of an active UWB ranging session.
2. Ranging Session Suspend Response Message ID SHALL NOT be sent without receiving the corresponding Ranging Session Suspend Request Message ID.
3. Ranging Session Resume Response Message ID SHALL NOT be sent without receiving the corresponding Ranging Session Resume Request Message ID.
4. Ranging Session Resume Request Message ID SHALL NOT be sent in the presence of an active UWB ranging session.
5. Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID SHALL NOT be sent in the presence of an active UWB ranging session.
6. An active UWB ranging session SHALL be suspend before sending Ranging Session Resume Request Message ID.
7. An active UWB ranging session SHALL be suspended before sending Ranging Message ID carrying Initiate Ranging Session Resume Attribute ID.
8. Ranging Message ID carrying Initiate Ranging Session Resume Later Attribute ID SHALL NOT be sent without receiving the corresponding Ranging Session Resume Request Message ID.

11.9.4.3 Miscellaneous

1. AP_RS Message ID SHALL NOT be sent without receiving the corresponding AP_RQ Message ID.
2. Reception of Time Sync Message ID SHALL NOT have any effect on an ongoing responseTimeout countdown at the Reader.
3. Reception of Reader Status Changed Message ID SHALL NOT have any effect on an ongoing responseTimeout countdown at the User Device.

11.10 Time Synchronization

The Bluetooth LE and UWB time synchronization functionality of Aliro is defined in section 19.4 in [2]. The time synchronization Procedure 0 and Procedure 1 SHALL be mandatory for the User Device. This means the User Device SHALL indicate the support of the Time Synchronization Procedure 0 and Time Synchronization Procedure 1 features inside the Selected ALIRO BLE UWB Protocol Version Characteristics (see Table 11-8 and Table 11-9). The time synchronization support is strongly RECOMMENDED for the Reader due to performance gains such as improved latency and power saving benefit for the Reader.

11.11 Considerations while referring to Digital Key Specification

Considerations while referring to [2] are listed below.

1. Without loss of generality, occurrences of ‘vehicle’, ‘Device’ and ‘DK’ in section 19.4 in [2] can be replaced with ‘Reader’, ‘User Device’, and ‘Aliro’, respectively for the purposes of this specification.
2. Without loss of generality replace ‘RSS-RS / RR-RS’ with ‘Ranging Session Response’ and ‘Ranging Resume Response’, respectively in section 19.4 in [2].
3. CRR-RS message in 19.4 in [2] is not applicable to in Aliro.
4. The Time Sync Message and attributes used in section 19.4 in [2] are formatted as described in 11.7.4.2 in this specification.
5. Replace Ranging session setup in 19.4.5 in [2] with ranging session setup described in 12.1.4 in this specification.
6. The terms ‘SE’, ‘Secure Element’, ‘applet’, ‘DK applet’, mentioned in [2], are only relevant for the CCC specification. This version of the specification does not define requirements on the secure environment, such as Secure Element, to be used for Aliro protocol execution. Having a secure environment is deemed a good security practice, more details can be found in section 16.

12 UWB Interface

12.1 UWB MAC and Channel Access

The MAC layer of the UWB-related functionality of Aliro is defined in section 20 in [2]. Here we describe the extensions to section 20 in [2] that are relevant to Aliro. See section 11.11 for considerations while referring to [2].

12.1.1 MAC Protocol

One or two ranging rounds out of all the ranging rounds per ranging block of a ranging session are used for the UWB ranging procedure. The number of ranging rounds out of all the ranging rounds per ranging block of a ranging session that are used for UWB ranging procedure depends on the responder-device (i.e., the Reader) design/implementation and SHALL be chosen by the responder-device during the ranging session setup. Two ranging rounds per ranging block enable "in front of" and "behind the Reader" detection by the Reader. If a Reader can perform this detection in first round, the second round is optional. However, if a Reader requires second ranging round for this detection, the User Device MUST support it. A Reader MAY optionally support two ranging rounds per ranging block while a User Device SHALL support two ranging rounds per ranging block. During the UWB ranging session setup (see section 12.1.4), the Reader indicates to the User Device in MAC Mode Attribute ID (see section 11.7.2.1.16) whether one or two ranging rounds per block will be used in the ranging session. Additionally, the responder-device SHALL be responsible for mapping responders (at the responder-device) to the appropriate ranging rounds.

A general description of the UWB Access MAC protocol is shown in Figure 12-1. In general, each of the ranging rounds that are used for UWB ranging procedure in the k-th ranging session SHALL be assigned to two different hopping sequences $H^k = \{h_0^k, h_1^k, \ldots, h_i^k\}$ and $F^k = \{f_0^k, \ldots\ldots, f_i^k\}$. Section 17 shows computation for the hopping sequence H^k. These two hopping sequences SHALL not be identical, that is $h_i^k \neq f_i^k \; \forall \; i$. In this specification, we ensure that by having the hopping sequences being an offset from one another, that is: $f_i^k = h_i^k + O^k \;\; \forall \; i$, where O^k is a session specific non-zero offset. The value of O^k (in number of rounds) depends on the responder-device design/implementation and SHALL be chosen by the responder-device during the ranging session setup and shall remain constant for the entire ranging session.

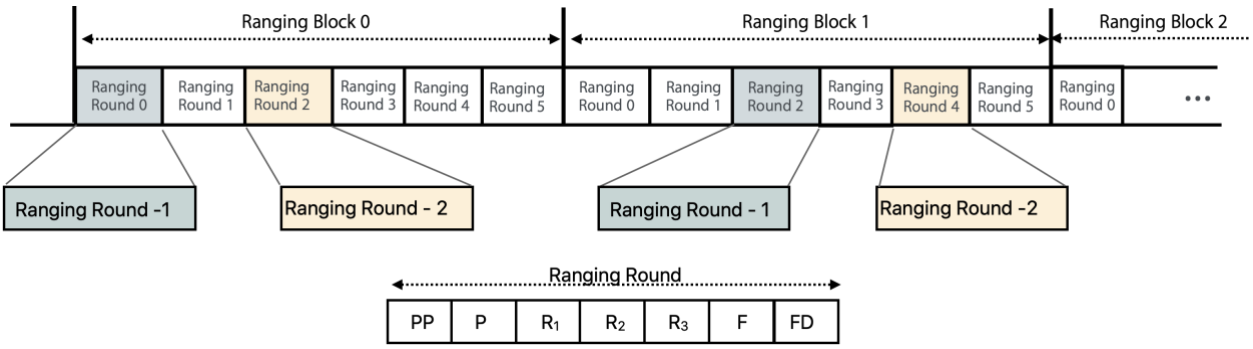


Figure 12-1 – General UWB Access MAC Protocol

12.1.2 Ranging Exchange Sequence

The UWB ranging procedure in a ranging round uses the ranging exchange sequence described in section 20.5 in [2] with the following additional considerations.

In the Pre-Poll packet

1. $Hop\_Flag_p^k(i)$: Hopping mode as set from the previous ranging exchange for the p-th ranging round (p =1 for the first ranging round and p = 2 for the second ranging round in the i-th ranging block in the k-th ranging session. See section 12.1.3 for details of how the hopping flag is set. If current ranging round is the first ranging round after the start of the ranging session, then

$Hop\_Flag_1^k(i=0)$ is set to '0' and $Hop\_Flag_2^k(i=0)$ is set to '0'. Note that this field is only relevant if the hopping mode is set to "adaptive hopping".

2. $Round\_Idx_p^k(i)$: p-th ranging round index (p = 1 for the first ranging round and p = 2 for the second ranging round) in the (i+1)-th ranging block in the k-th ranging session as set in Final_Data packet of the previous ranging exchange (i-th ranging block). If the current ranging round is the first round in the ranging session, then $Round\_Idx_1^k(i=0)$ SHALL be set to '0' and $Round\_Idx_2^k(i=0)$ SHALL be set to O^k .

In the Final_Data packet

1. $Hop\_Flag_p^k(i+1)$: Hopping flag to be used in the p-th ranging round (p = 1 for the first ranging round and p = 2 for the second ranging round) in ranging block i+1. Note that this field is only relevant if the hopping configuration is set to adaptive hopping.
2. $Round\_Idx_p^k(i+1)$: p-th ranging round index (p = 1 for the first ranging round and p = 2 for the second ranging round) of the next ranging exchange in ranging block i+1.
3. $STS\_Index^k(i, Round\_Idx_p^k(i), FINAL)$: STS index of the preceding FINAL message.
4. $Final\_Time\_Stamp_p^k(i)$: the time stamp for the Final POLL message. This time stamp SHALL be calculated as the difference between the RMARKER of the initiator POLL message and the RMARKER of the initiator Final POLL message.

12.1.3 Hopping Flag and Round Index Determination

As stated above, the initiator in the k-th ranging session in any given RAN SHALL start the UWB ranging procedure in the first ranging round (ranging round number 0) in the first ranging block (ranging block number 0) by default if there is only one ranging round per block and if there are two ranging rounds per block, then the initiator SHALL start the UWB ranging in ranging rounds 0 and O^k in the first ranging block. This assumes that the responder-device has either achieved block synchronization by OOB method or, if not, is permanently listening for Pre-Poll messages.

At the initiator, and assuming no resource conflict occurs, the $Hop\_Flag_p^k(i+1)$ and $Round\_Idx_p^k(i+1)$ for the next ranging block (ranging block i+1) will depend on the hopping mode selected during ranging session setup as follows:

- If the hopping mode is set to "no hopping", then the initiator SHALL continue to use the same ranging round in ranging block i and $Round\_Idx_p^k(i+1)$ is set as:

$$Round\_Idx_1^k(i+1) = Round\_Idx_1^k(i) = 0,$$
$$Round\_Idx_2^k(i+1) = Round\_Idx_2^k(i) = O^k,$$

and $Hop\_Flag_p^k(i+1)$ is set to 0. Note that in this case, $Hope\_Flag_p^k(i+1)$ is irrelevant to the receiver and SHALL be ignored.

- If the hopping mode is set to "continuous hopping", the initiator uses the h_{i+1}^k and f_{i+1}^k ranging rounds in the next ranging block

$$Round\_Idx_1^k(i+1) = h_{i+1}^k \text{ and } Round\_Idx_2^k(i+1) = f_{i+1}^k$$

And $Hop\_Flag_p^k(i+1)$ is set to 1. Again, $Hop\_Flag_p^k(i+1)$ is irrelevant to the receiver and SHALL be ignored.

- If the hopping mode is set to “adaptive hopping”, then at the initiator, the $Hop\_Flag_p^k(i+1)$ and $Round\_Idx_p^k(i+1)$ for the first and second ranging rounds in the next ranging block (ranging block *i*+1) SHALL be set as follows:

1. For the first ranging round:

   If the initiator determines that the round is clean, i.e., no interference and ranging in the current round is successful, the initiator SHALL stay in the current round and set.

   $Hop\_Flag_1^k(i+1) = 0$ and $Round\_Idx_1^k(i+1) = h_i^k$

   If the initiator determines that there is some interference on the current round or if the ranging is not successful, the initiator SHALL hop to a different round:

   $Hop\_Flag_1^k(i+1) = 1$ and $Round\_Idx_1^k(i+1) = h_{i+1}^k$

   The initiator SHALL send the $Hop\_Flag_1^k(i+1)$ and $Round\_Idx_1^k(i+1)$ for next ranging block to the responder-device as part of the payload packet carrying time stamps *Final_Data* message at the end of the ranging sequence of the first ranging round.

2. For the second ranging round:

   If the initiator did not turn on hopping in the first ranging round and the initiator determines that in the second round, there is no/little interference and ranging in the second ranging round is successful, the initiator SHALL stay in the current round and set.

   $Hop\_Flag_2^k(i+1) = 0$ and $Round\_Idx_2^k(i+1) = f_i^k$

   If the initiator determines that there is some interference in the second ranging round or if the second ranging round is not successful, the initiator SHALL hop to a different round:

   $Hop\_Flag_2^k(i+1) = 1$ and $Round\_Idx_2^k(i+1) = f_{i+1}^k$

   The initiator SHALL send the $Hop\_Flag_2^k(i+1)$ and $Round\_Idx_2^k(i+1)$ for next ranging block to the responder-device as part of the payload packet carrying time stamps *Final_Data* message at the end of the ranging sequence of the second ranging round.

   Note that in each ranging round, the interference present might be different and either of the ranging rounds would success/failure independently. Therefore, at the initiator the final hopping flag and round index for the next ranging block are

   $Hop\_Flag^k(i+1) = Hop\_Flag_1^k(i+1)$ **OR** $Hop\_Flag_2^k(i+1)$

   $Round\_Idx_1^k(i+1) = h_i^k$, $Round\_Idx_2^k(i+1) = f_i^k$, if Hop_Flag^k(i+1) = 0

   $Round\_Idx_1^k(i+1) = h_{i+1}^k$, $Round\_Idx_2^k(i+1) = f_{i+1}^k$, if Hop_Flag^k(i+1) = 1

At the responder-device, the hopping and round indices for subsequent ranging rounds SHALL be resolved as follows:

if either $Final\_Data_1$ or $Final\_Data_2$ has not be received

{

$$Hop\_Flag_1^k(i+1) = Hop\_Flag_2^k(i+1) = 1$$

$$Round\_Idx_1^k(i+1) = h_{i+1}^k$$

$$Round\_Idx_2^k(i+1) = f_{i+1}^k$$

}

else

{

$Hop\_Mode^k(i+1) = (Final\_Data_1.Hop\_Flag$ **OR** $Final\_Data_2.Hop\_Flag)$

if $(Hop\_Mode^k(i+1) == 0)$

{

set $Round\_Idx_1^k(i+1) = h_i^k; Round\_Idx_2^k(i+1) = f_i^k$

}

elseif $(Hop\_Mode^k(i+1) == 1)$

{

set $Round\_Idx_1^k(i+1) = h_{i+1}^k; Round\_Idx_2^k(i+1) = f_{i+1}^k$

}

}

12.1.4 Ranging Session Setup

The UWB ranging session parameters are agreed upon between the initiator and the responder through ranging session setup shown in Figure 12-2 and described below.

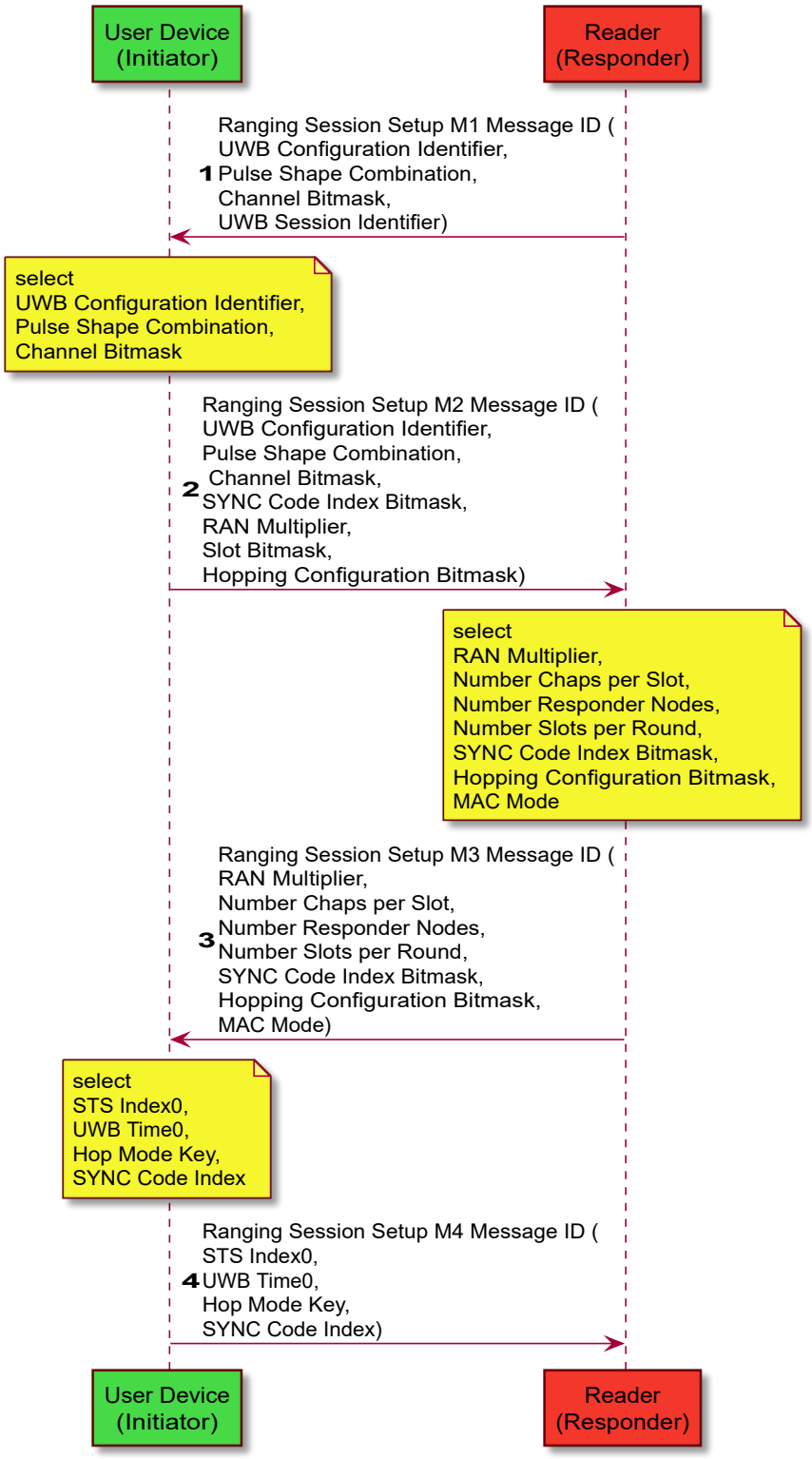


Figure 12-2 – UWB ranging session setup

The responder-device initiates ranging capability exchange by sending Ranging Session Setup M1 Message ID (see section 11.7.2.2) to the initiator. The Ranging Session Setup M1 Message ID includes following Attribute IDs (see Table 11-13):

1. UWB Configuration Identifier,
2. Pulse Shape Combination,
3. Channel Bitmask: $\{CH\_IDX\}_{Responder}$ list of available UWB RF channels.
4. UWB Session Identifier: Identifier of the current ranging session.

The Initiator responds with Ranging Session Setup M2 Message ID (see section 11.7.2.3) on receiving the Ranging Session Setup M1 Message ID from the Responder-device. The Ranging Session Setup M2 Message ID includes the following Attribute IDs:

1. UWB Configuration Identifier: indicate the selected UWB configuration identifier for the ranging session.
2. Pulse Shape Combination: User Device selected single pulse shape combination from the list of common supported ones between the User Device and the Reader.
3. Channel Bitmask: CH_IDX of the selected UWB RF channel.
4. SYNC Code Index Bitmask $\{SYC\_IDX\}_{Initiator}$: list of available UWB preamble sequences that the responder-device MAY choose from.
5. RAN Multiplier: RAN multiplier, sets the minimum ranging block duration for the *k*-th ranging session that can be supported by the initiator (i.e., sets the maximum ranging frequency that can be supported by the initiator for this ranging session)

$$T^k_{Block\_RAN} = N^k_{RAN} \times T_{Block\_Min} = N^k_{RAN} \times 96\,ms$$

$$f^k_{Ranging\_RAN} = \frac{1}{T^k_{Block\_RAN}} = \frac{10.416667}{N^k_{RAN}}\ \text{Hz}$$

6. Slot Bitmask $(N_{Chap\_per\_Slot})_{Initiator}$: list of slot durations supported by the initiator for the session expressed as specified in section 19.3.1.4 in [2] and Table G-1 in [2].
7. Hopping Configuration Bitmask $\{H\_Config\_Seq\}_{Initiator}$ bit mask indicating the hopping configuration (no hopping, continuous, adaptive) and hopping sequences supported by the device.

Responder-device responds with Ranging Session Setup M3 Message ID (see section 11.7.2.4) to the initiator after receiving Ranging Session Setup M2 Message ID. The Ranging Session Setup M3 Message ID includes the following Attribute ID:

1. RAN Multiplier: selected N^k_{RAN} for the ranging session: the responder-device selects an value that is greater than or equal to the value sent by the initiator to achieve a desired ranging interval of that is an integer multiple of T_{Block_Min}.

   $N^k_{RAN\_S} = \frac{T^k_{Block}}{T_{Block\_Min}}$, $N^k_{RAN\_S} \geq N^k_{RAN}$, and $f^k_{Ranging} = \frac{1}{T^k_{Block}} = \frac{10.416667}{N^k_{RAN\_S}}$ Hz

2. Number Chaps per Slot $N^k_{Chap\_per\_Slot}$: shortest slot duration that is common between slot durations supported by initiator $\{N^k_{Chap\_per\_Slot}\}_{Initiator}$ and slot durations supported by the responder-device $\{N_{Chap\_per\_Slot}\}_{Responder}$:

$$N^k_{Chap\_per\_Slot} = min\left(\{N_{Chap\_per\_Slot}\}_{Initiator} \cap \{N_{Chap\_per\_Slot}\}_{Responder}\right)$$

3. Number Responder Nodes $N^k_{Responder}$: Number of responder nodes participating in a ranging round in this ranging session as selected by the responder-device.
4. Number Slots per Round $N^k_{Slot\_per\_Round}$: responder-device selects the number of slots that is greater than or equal to $(N^k_{Responder} + 4)$ out of all possible values of slots corresponding to $N^k_{Chap\_per\_Slot}$ (see Table G-1 in [2] for details).
5. SYNC Code Index Bitmask $\{SYC\_IDX\}_{Responder}$: list of UWB preamble sequences that the responder-device selected to use. This set is a subset of the list sent by the transmitter.
6. Hopping Configuration Bitmask: $\{H\_Config\_Seq\}_{Responder}$ bit mask indicating the hopping configuration and hopping sequences selected by the Reader.
7. MAC Mode: responder-device indicates number of ranging rounds out of all the ranging rounds in a ranging block that are used for UWB ranging procedure. It also indicates ranging round offset (in ranging rounds) between the two ranging rounds O^k, $1 \leq O^k \leq N^k_{Round} - 1$ in a ranging block that are used for UWB ranging procedure.

Initiator responds with Ranging Session Setup M4 Message ID (see section 11.7.2.5) to the Responder-device after receiving Ranging Session Setup M3 Message ID. The Ranging Session Setup M4 Message ID includes the following Attribute IDs:

1. STS Index0: starting STS index.
2. UWB Time0: starting time reference on UWB Device Clock of ranging session.
3. Hop Mode Key ($HOP\_Key\_RW^k$): Key to generate default hopping sequence.
4. SYNC Code Index: selected SYNC code index.

The Initiator selects the number of rounds per block for the ranging session as

$$N^k_{Round} = \frac{288 \times N^k_{RAN\_S}}{N^k_{Chap\_per\_Slot} \times N^k_{Slot\_per\_Round}}$$

See Appendix G.1 in [2] for a list of all valid/possible number of rounds for different combinations of $(N^k_{Chap\_per\_Slot}, N^k_{Slot\_per\_Round})$ for each 96 ms of block duration.

12.1.5 UWB MAC Configuration

The Vendor OUI in the Vendor Specific Header IE of the MHR Field of the SP0 packet SHALL be set to 0x4A191B. The value 0x4A191B is CSA Company Identifier per IEEE registration authority.

12.2 UWB PHY

The physical layer of the UWB-related functionality of Aliro is defined in section 21 in [2]. See section 11.11 for considerations while referring to [2].

12.3 UWB Security

The security requirements to the UWB-related functionality of Aliro are defined in section 22.1 and 22.2 in [2]. See section 11.11 for considerations while referring to [2].

13 Appendix with certificate requirements

13.1 Credential Issuer certificate requirements

The Credential Issuer certificate SHALL be issued by Credential Issuer CA.

The Credential Issuer certificate SHALL be implemented according to the following requirements:

The certificate is a DER encoded certificate of type “v3” according to [12].

The following fields SHALL be present, other fields MAY be present:

- Version
- Serial number
- Issuer
- Validity
  - Not before
  - Not after
- Subject
- Subject public key info
  - Algorithm
  - Parameters
  - subjectPublicKey
- Extensions
  - Authority key identifier extension
    - keyIdentifier
  - Key usage extension
- Signature algorithm
- Signature value

The authority key identifier extension SHALL be non-critical.

The key usage extension SHALL be critical with only the digital signature bit set to 1.

The Basic Constraint extension MAY be present, either critical or non-critical. Other extensions MAY be present if they are non-critical as defined in section 4.2 of [12].

The signature algorithm SHALL be ECDSA-with-SHA256.

Key authority public key SHALL be P-256.

13.2 Reader certificate requirements

The Reader certificate SHALL be issued by the Reader System Issuer CA. The Reader certificate SHALL be implemented such that it can be compressed as per profile0000, this corresponds to the following requirements certificate:

The Reader certificate SHALL be a DER encoded certificate of type “v3” according to [12].

The following fields SHALL be present, other fields SHALL NOT be present:

- Version
- Serial number
- Issuer
- Validity
  - Not before
  - Not after
- Subject
- Subject public key info
  - Algorithm
  - Parameters
  - subjectPublicKey
- Extensions
  - Authority key identifier
    - keyIdentifier
  - Basic constraints
  - Key usage extension
- Signature algorithm
- Signature value

13.3 Reader certificate compression

This specification defines a compressed certificate format called profile0000. A profile0000 data structure can be uncompressed into an RFC5280 compliant X.509 certificate.

The profile0000 data structure only conveys a subset of the fields normally present in an X509 certificate, the remaining fields are implicit and have to be reconstructed by the verifier.

The Profile0000 data structure SHALL be encoded as DER according to the scheme defined below to be used in the AUTH1 command or LOAD CERT command.

The profile0000 data structure is defined using an ASN.1 scheme as follow:

```
Schema DEFINITIONS IMPLICIT TAGS ::=
BEGIN
  Profile0000 ::= SEQUENCE
  {
     profile   OCTET STRING (SIZE (2)),
     data      Profile0000Data
  }

  Profile0000Data ::= SEQUENCE
  {
     serialNumber   [0] OCTET STRING (SIZE (1..20)) OPTIONAL,
     issuer         [1] OCTET STRING (SIZE (1..32)) OPTIONAL,
     notBefore      [2] OCTET STRING (SIZE (13..15)) OPTIONAL,
     notAfter       [3] OCTET STRING (SIZE (13..15)) OPTIONAL,
     subject        [4] OCTET STRING (SIZE (1..32)) OPTIONAL,
```

```
    publicKey       [5] OCTET STRING,
    signature       [6] OCTET STRING
  }
END
```

profile: this mandatory field contains the 2 bytes profile number, since only one profile is currently supported this field contains the value 0x0000 pointing to profile0000.

data: this mandatory field contains the fields to be inserted in the uncompressed X509 certificate.

serialNumber: this OPTIONAL field contains the bytes to be inserted in the ASN.1 INTEGER encoding the serialNumber of the uncompressed X509 certificate. If omitted, the following default bytes are used: 0x01.

issuer: this OPTIONAL field contains the bytes to be inserted in the ASN.1 UTF8String encoding the Issuer Common Name of the uncompressed X509 certificate. If omitted, the following default bytes are used: 0x697373756572.

notBefore: this OPTIONAL field contains the bytes to be inserted in the ASN.1 UTCTime or GeneralizedTime, encoding the notBefore validity of the uncompressed X509 certificate. Note that UTCTime and GeneralizedTime use a different tag in the encoded certificate. If omitted, the following default bytes are used: 0x323030313031303030303030305A.

notAfter: this OPTIONAL field contains the bytes to be inserted in the ASN.1 UTCTime or GeneralizedTime, encoding the notAfter validity of the uncompressed X509 certificate. Note that UTCTime and GeneralizedTime use a different tag in the encoded certificate. If omitted, the following default bytes are used: 0x343930313031303030303030305A.

subject: this OPTIONAL field contains the bytes to be inserted in the ASN.1 UTF8String encoding the Subject Common Name of the uncompressed X509 certificate. If omitted, the following default bytes are used: 0x7375626A656374.

publicKey: this mandatory field contains the bytes to be inserted in the ASN.1 BIT STRING encoding the subject public key of the uncompressed X509 certificate, this byte sequence includes the first byte representing the number of unused bits in a DER encoded BIT STRING.

signature: this mandatory field contains the bytes to be inserted in the ASN.1 BIT STRING encoding the signature of the uncompressed X509 certificate, this byte sequence includes the first byte representing the number of unused bits in a DER encoded BIT STRING.

The reference X509 certificate to be used for profile0000 is defined below. Only the fields mentioned above and the content of the OCTET STRING present in the Authority Key Identifier extension can differ from this definition.

The length fields of the uncompressed certificate DER structure SHALL be generated wherever necessary by the verifier during certificate decompression.

The OCTET STRING present in the Authority Key Identifier extension SHALL be generated as per RFC5280 recommendation using the following method: The keyIdentifier is composed of the 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey as uncompressed point (excluding the tag, length, and number of unused bits).

Reference x509 for profile0000:

```
308201513081f9a003020102020101300a06082a8648ce3d0403023011310f300d06035504030c066973737565
72301e170d323030313031303030303030305a170d343930313031303030303030305a30123110300e06035504030c
077375626a65637430593013060 72a8648ce3d020106082a8648ce3d030107034200048422 42f6182ba1c1138d
32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243a
c8544a665cb951422fa341303f301f0603551d23041830168014 2318e55671f08eae212142a817720fb817ee93
bf300c0603551d130101ff04023000300e0603551d0f0101ff040403020780300a06082a8648ce3d0403020347
00304402201610f6e9fbc7ddfd46bb9b585627285daf676eb3a950d99ed6d462763ef5fb7102202208fd466e06
a77327865c50430e73f808389644351b390b92eee853eacb2619
```

13.3.1 Compression Steps

1. Verify the input X509 certificate conforms with the reference X509 certificate.
2. Extract the profile0000 bytes arrays mentioned above (serialNumber, issuer, notBefore…)
3. Discard the byte array when the value is equal to the default value
4. Generate Profile0000 DER encoded data structure as per ASN.1 scheme above
5. The compressed certificate is ready to be provisioned on the target system

13.3.2 Decompression Steps

1. Verify the Profile0000 data structure is compliant with the defined ASN.1 scheme for this profile
2. Extract the byte arrays from Profile0000
3. Generate the OCTET STRING for the Authority Key Identifier extension using the public key that will be used to verify this certificate
4. Insert the extracted and generated fields into the reference X509 certificate and adjust lengths
5. The uncompressed certificate signature is ready to be verified

14 Appendix with Example Flow Diagrams

14.1 Reader certificate compression examples

```
##
## Demo 1 using reference cert as input
##
## Reader public key
04842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc340
1c3a4f4e5a59251d45243ac8544a665cb951422f
## Reader  private key
308187020100301306072a8648ce3d020106082a8648ce3d030107046d306b02010104201a39e361b0db1915c2
510bd92f3dbeb319ed68b16a0294347629d2e4becdb599a14403420004842242f6182ba1c1138d32b77fb9f7f3
7b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb9
51422f
## Issuer public key
04793e3a8f20428d54e7318046d75d05a8737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a9
7bb5bfa9e623a3cc886b09dc13d53ef0da7de7bd
## Issuer private key
308187020100301306072a8648ce3d020106082a8648ce3d030107046d306b02010104204b45df37a327a31303
113f9965d14de94f025f881515e13034a3d8a9ac47e43ea14403420004793e3a8f20428d54e7318046d75d05a8
737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a97bb5bfa9e623a3cc886b09dc13d53ef0da
7de7bd
## Input X509 Certificate
308201523081f9a003020102020101300a06082a8648ce3d0403023011310f300d06035504030c066973737565
72301e170d323030313031303030303030305a170d343930313031303030303030305a30123110300e06035504030c
077375626a656374305930130 6072a8648ce3d020106082a8648ce3d03010703420004842242f6182ba1c1138d
32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243a
c8544a665cb951422fa341303f301f0603551d230418301680142318e55671f08eae212142a817720fb817ee93
bf300c0603551d130101ff04023000300e0603551d0f0101ff040403020780300a06082a8648ce3d0403020348
00304502210 08720a2f08626d56b7814b7e5bbe04381e1834cf9a2a5d4c85c76783607a22cc60220236a4b757c
d497c8570e84fa3221be99f6c78cc7cbc71d7328aa99be03f1eccf
##compressing  cert..##
serial number: 01
serialNumber field contains default value -> remove from profiled cert
issuer: 697373756572
issuer field contains default value -> remove from profiled cert
not before: 32303031303130303030305a
notBefore field contains default value -> remove from profiled cert
not after: 343930313031303030305a
notAfter field contains default value -> remove from profiled cert
subject: 7375626a656374
subject field contains default value -> remove from profiled cert
public key:
0004842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3
401c3a4f4e5a59251d45243ac8544a665cb951422f
signature:
00304502210 08720a2f08626d56b7814b7e5bbe04381e1834cf9a2a5d4c85c76783607a22cc60220236a4b757c
d497c8570e84fa3221be99f6c78cc7cbc71d7328aa99be03f1eccf
190 bytes saved by compression
## Compressed Certificate
308195040200003 0818e85420004842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb
36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb951422f864800304502210 08720a2f0
8626d56b7814b7e5bbe04381e1834cf9a2a5d4c85c76783607a22cc60220236a4b757cd497c8570e84fa3221be
99f6c78cc7cbc71d7328aa99be03f1eccf
##uncompressing cert..##
issuer_public_key:
04793e3a8f20428d54e7318046d75d05a8737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a9
7bb5bfa9e623a3cc886b09dc13d53ef0da7de7bd
aki extension: 301680142318e55671f08eae212142a817720fb817ee93bf
serialNumber not present in profile --> use default value
issuer not present in profile --> use default value
notBefore not present in profile --> use default value
notAfter not present in profile --> use default value
subject not present in profile --> use default value
## Uncompressed Certificate
308201523081f9a003020102020101300a06082a8648ce3d0403023011310f300d06035504030c066973737565
72301e170d323030313031303030303030305a170d343930313031303030303030305a30123110300e06035504030c
077375626a656374305930130 6072a8648ce3d020106082a8648ce3d03010703420004842242f6182ba1c1138d
32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243a
c8544a665cb951422fa341303f301f0603551d230418301680142318e55671f08eae212142a817720fb817ee93
```

```
bf300c0603551d130101ff04023000300e0603551d0f0101ff040403020780300a06082a8648ce3d0403020348
0030450221008720a2f08626d56b7814b7e5bbe04381e1834cf9a2a5d4c85c76783607a22cc60220236a4b757c
d497c8570e84fa3221be99f6c78cc7cbc71d7328aa99be03f1eccf

##
## Demo 2 using cert with customized fields
##
## Reader public key
04842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc340
1c3a4f4e5a59251d45243ac8544a665cb951422f
## Reader  private key
308187020100301306072a8648ce3d020106082a8648ce3d030107046d306b02010104201a39e361b0db1915c2
510bd92f3dbeb319ed68b16a0294347629d2e4becdb599a14403420004842242f6182ba1c1138d32b77fb9f7f3
7b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb9
51422f
## Issuer public key
04793e3a8f20428d54e7318046d75d05a8737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a9
7bb5bfa9e623a3cc886b09dc13d53ef0da7de7bd
## Issuer private key
308187020100301306072a8648ce3d020106082a8648ce3d030107046d306b02010104204b45df37a327a31303
113f9965d14de94f025f881515e13034a3d8a9ac47e43ea14403420004793e3a8f20428d54e7318046d75d05a8
737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a97bb5bfa9e623a3cc886b09dc13d53ef0da
7de7bd
## Input X509 Certificate
308201643082010aa003020102020604278ba9fd71300a06082a8648ce3d040302301d311b3019060355040 30c
12637573746f6d20697373756572206e616d65301e170d323033013031303030303030305a170d32353035303530
30303030305a30123110300e06035504030c077375626a656374305930130 6072a8648ce3d020106082a8648ce
3d030107034200048422 42f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f9
1a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb951422fa341303f301f0603551d2304183016801423
18e55671f08eae212142a817720fb817ee93bf300c0603551d130101ff04023000300e0603551d0f0101ff0404
03020780300a06082a8648ce3d040302034800304502206080fed25cf442226d5017c0e3f5f929ff5cbd18bfa7
53cbd876c02f0a8abbb4022100bc3e990a9cec57b1c1717fdeb6aab55cece7c96fff47bf5a7236accfb378347e
##compressing  cert..##
serial number: 04278ba9fd71
issuer: 637573746f6d20697373756572206e616d65
not before: 32303031303130303030305a
notBefore field contains default value -> remove from profiled cert
not after: 32353035303530303030305a
subject: 7375626a656374
subject field contains default value -> remove from profiled cert
public key:
0004842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3
401c3a4f4e5a59251d45243ac8544a665cb951422f
signature:
00304502206080fed25cf442226d5017c0e3f5f929ff5cbd18bfa753cbd876c02f0a8abbb4022100bc3e990a9c
ec57b1c1717fdeb6aab55cece7c96fff47bf5a7236accfb378347e
165 bytes saved by compression
## Compressed Certificate
3081c0040200003081b9800604278ba9fd718112637573746f6d20697373756572206e616d65830d3235303530
353030303030305a854200048422 42f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb3649
0a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb951422f8648003045022060 80fed25cf442
226d5017c0e3f5f929ff5cbd18bfa753cbd876c02f0a8abbb4022100bc3e990a9cec57b1c1717fdeb6aab55cec
e7c96fff47bf5a7236accfb378347e
##uncompressing cert..##
issuer_public_key:
04793e3a8f20428d54e7318046d75d05a8737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a9
7bb5bfa9e623a3cc886b09dc13d53ef0da7de7bd
aki extension: 301680142318e55671f08eae212142a817720fb817ee93bf
serialNumber: 04278ba9fd71
serialNumber present in profile --> use in cert
issuer: 637573746f6d20697373756572206e616d65
issuer present in profile --> use in cert
notBefore not present in profile --> use default value
notAfter: 32353035303530303030305a
notAfter present in profile --> use in cert
subject not present in profile --> use default value
## Uncompressed Certificate
308201643082010aa003020102020604278ba9fd71300a06082a8648ce3d040302301d311b3019060355040 30c
12637573746f6d20697373756572206e616d65301e170d323033013031303030303030305a170d32353035303530
30303030305a30123110300e06035504030c077375626a656374305930130 6072a8648ce3d020106082a8648ce
3d030107034200048422 42f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f9
1a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb951422fa341303f301f0603551d2304183016801423
18e55671f08eae212142a817720fb817ee93bf300c0603551d130101ff04023000300e0603551d0f0101ff0404
03020780300a06082a8648ce3d040302034800304502206080fed25cf442226d5017c0e3f5f929ff5cbd18bfa7
```

```
53cbd876c02f0a8abbb4022100bc3e990a9cec57b1c1717fdeb6aab55cece7c96fff47bf5a7236accfb378347e

##
## Demo 3 using cert with customized fields
##
## Reader public key
04842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc340
1c3a4f4e5a59251d45243ac8544a665cb951422f
## Reader  private key
308187020100301306072a8648ce3d020106082a8648ce3d030107046d306b02010104201a39e361b0db1915c2
510bd92f3dbeb319ed68b16a0294347629d2e4becdb599a14403420004842242f6182ba1c1138d32b77fb9f7f3
7b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb9
51422f
## Issuer public key
04f47eb42a771052580c086efdaaa3084aa3ff7a67ce23393a0373c63487df1a637d1fb34b2d2e7d5c8f92097a
0619b5c5cc6c5850af74c019ebbec4273358aa94
## Issuer private key
308187020100301306072a8648ce3d020106082a8648ce3d030107046d306b020101042086b9e3843d949890fd
50e49c8542db575bac41d344f17588ddafe4535521ce55a14403420004f47eb42a771052580c086efdaaa3084a
a3ff7a67ce23393a0373c63487df1a637d1fb34b2d2e7d5c8f92097a0619b5c5cc6c5850af74c019ebbec42733
58aa94
## Input X509 Certificate
308201513081f9a003020102020101300a06082a8648ce3d0403023011310f300d06035504030c066973737565
72301e170d323030313031303030303030305a170d323530353035303030303030305a30123110300e06035504030c
077375626a656374305930130 6072a8648ce3d020106082a8648ce3d030107034200048422 42f6182ba1c1138d
32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243a
c8544a665cb951422fa341303f301f0603551d230418301680147faeab3831311eac3c8bdc7d49cd0f8b3f1a9c
2f300c0603551d130101ff04023000300e0603551d0f0101ff040403020780300a06082a8648ce3d0403020347
00304402205a15bb0cd0718e077815fe8c71ddb05378c89fbf5ae2f976f2b506fcc224fa0402201aae5e32782d
d979e71c8e1e6ba31054b121ac77933a4a7b3cf10e97cb64b9fe
##compressing  cert..##
serial number: 01
serialNumber field contains default value -> remove from profiled cert
issuer: 697373756572
issuer field contains default value -> remove from profiled cert
not before: 323030313031303030303030305a
notBefore field contains default value -> remove from profiled cert
not after: 323530353035303030303030305a
subject: 7375626a656374
subject field contains default value -> remove from profiled cert
public key:
0004842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3
401c3a4f4e5a59251d45243ac8544a665cb951422f
signature:
00304402205a15bb0cd0718e077815fe8c71ddb05378c89fbf5ae2f976f2b506fcc224fa0402201aae5e32782d
d979e71c8e1e6ba31054b121ac77933a4a7b3cf10e97cb64b9fe
175 bytes saved by compression
## Compressed Certificate
3081a30402000030819c830d323530353035303030303030305a85420004842242f6182ba1c1138d32b77fb9f7f3
7b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb9
51422f864700304402205a15bb0cd0718e077815fe8c71ddb05378c89fbf5ae2f976f2b506fcc224fa0402201a
ae5e32782dd979e71c8e1e6ba31054b121ac77933a4a7b3cf10e97cb64b9fe
##uncompressing cert..##
issuer_public_key:
04f47eb42a771052580c086efdaaa3084aa3ff7a67ce23393a0373c63487df1a637d1fb34b2d2e7d5c8f92097a
0619b5c5cc6c5850af74c019ebbec4273358aa94
aki extension: 301680147faeab3831311eac3c8bdc7d49cd0f8b3f1a9c2f
serialNumber not present in profile --> use default value
issuer not present in profile --> use default value
notBefore not present in profile --> use default value
notAfter: 323530353035303030303030305a
notAfter present in profile --> use in cert
subject not present in profile --> use default value
## Uncompressed Certificate
308201513081f9a003020102020101300a06082a8648ce3d0403023011310f300d06035504030c066973737565
72301e170d323030313031303030303030305a170d323530353035303030303030305a30123110300e06035504030c
077375626a656374305930130 6072a8648ce3d020106082a8648ce3d030107034200048422 42f6182ba1c1138d
32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243a
c8544a665cb951422fa341303f301f0603551d230418301680147faeab3831311eac3c8bdc7d49cd0f8b3f1a9c
2f300c0603551d130101ff04023000300e0603551d0f0101ff040403020780300a06082a8648ce3d0403020347
00304402205a15bb0cd0718e077815fe8c71ddb05378c89fbf5ae2f976f2b506fcc224fa0402201aae5e32782d
d979e71c8e1e6ba31054b121ac77933a4a7b3cf10e97cb64b9fe

##
```

```
## Demo 4 using cert with max size fields
##
## Reader public key
04842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc340
1c3a4f4e5a59251d45243ac8544a665cb951422f
## Reader  private key
308187020100301306072a8648ce3d020106082a8648ce3d030107046d306b02010104201a39e361b0db1915c2
510bd92f3dbeb319ed68b16a0294347629d2e4becdb599a14403420004842242f6182ba1c1138d32b77fb9f7f3
7b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb9
51422f
## Issuer public key
04793e3a8f20428d54e7318046d75d05a8737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a9
7bb5bfa9e623a3cc886b09dc13d53ef0da7de7bd
## Issuer private key
308187020100301306072a8648ce3d020106082a8648ce3d030107046d306b02010104204b45df37a327a31303
113f9965d14de94f025f881515e13034a3d8a9ac47e43ea14403420004793e3a8f20428d54e7318046d75d05a8
737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a97bb5bfa9e623a3cc886b09dc13d53ef0da
7de7bd
## Input X509 Certificate
308201993082013fa00302010202145555555555555555555555555555555555555555300a06082a8648ce3d04
0302302b312930270603550403 0c20637573746f6d20697373756572206e616d652e2e2e2e2e2e2e2e2e2e2e
2e2e301e170d323030313032303030303030305a170d323530353035303030303030305a302b312930270603550403
0c20637573746f6d207375626a656374206e616d652e2e2e2e2e2e2e2e2e2e2e2e3059301306072a8648ce3d
020106082a8648ce3d030107034200048422 42f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188b
eadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb951422fa341303f301f0603551d
23041830168014 2318e55671f08eae212142a817720fb817ee93bf300c0603551d130101ff04023000300e0603
551d0f0101ff040403020780300a06082a8648ce3d0403020348003045022 06a552690283860fc94916ebbc92f
3510abb62a7a4729be57c2bde9fadf417e71022100c2385d82cfb33a357d5402f3e20fb271d0145b72b38a4a2b
4a6ebc6e14dd83b5
##compressing  cert..##
serial number: 55555555555555555555555555555555555555555
issuer: 637573746f6d20697373756572206e616d652e2e2e2e2e2e2e2e2e2e2e2e2e2e
not before: 323030313032303030303030305a
not after: 323530353035303030303030305a
subject: 637573746f6d207375626a656374206e616d652e2e2e2e2e2e2e2e2e2e2e2e2e2e
public key:
0004842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc3
401c3a4f4e5a59251d45243ac8544a665cb951422f
signature:
00304502206a552690283860fc94916ebbc92f3510abb62a7a4729be57c2bde9fadf417e71022100c2385d82cf
b33a357d5402f3e20fb271d0145b72b38a4a2b4a6ebc6e14dd83b5
139 bytes saved by compression
## Compressed Certificate
3082010e040200003082010680145555555555555555555555555555555555555555581206373746f6d206973
73756572206e616d652e2e2e2e2e2e2e2e2e2e2e2e2e2e820d323030313032303030303030305a830d3235303530
353030303030305a8420637573746f6d207375626a656374206e616d652e2e2e2e2e2e2e2e2e2e2e2e854200
04842242f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188beadb36490a7e95f91a4c162acfc340
1c3a4f4e5a59251d45243ac8544a665cb951422f864800304502206a552690283860fc94916ebbc92f3510abb6
2a7a4729be57c2bde9fadf417e71022100c2385d82cfb33a357d5402f3e20fb271d0145b72b38a4a2b4a6ebc6e
14dd83b5
##uncompressing cert..##
issuer_public_key:
04793e3a8f20428d54e7318046d75d05a8737eb6e074e5146a207bff62dae90e24039f372814a312c3cb82a5a9
7bb5bfa9e623a3cc886b09dc13d53ef0da7de7bd
aki extension: 301680142318e55671f08eae212142a817720fb817ee93bf
serialNumber: 55555555555555555555555555555555555555555
serialNumber present in profile --> use in cert
issuer: 637573746f6d20697373756572206e616d652e2e2e2e2e2e2e2e2e2e2e2e2e2e
issuer present in profile --> use in cert
notBefore: 323030313032303030303030305a
notBefore present in profile --> use in cert
notAfter: 323530353035303030303030305a
notAfter present in profile --> use in cert
subject: 637573746f6d207375626a656374206e616d652e2e2e2e2e2e2e2e2e2e2e2e2e2e
subject present in profile --> use in cert
## Uncompressed Certificate
308201993082013fa00302010202145555555555555555555555555555555555555555300a06082a8648ce3d04
0302302b312930270603550403 0c20637573746f6d20697373756572206e616d652e2e2e2e2e2e2e2e2e2e2e
2e2e301e170d323030313032303030303030305a170d323530353035303030303030305a302b312930270603550403
0c20637573746f6d207375626a656374206e616d652e2e2e2e2e2e2e2e2e2e2e2e2e2e3059301306072a8648ce3d
020106082a8648ce3d030107034200048422 42f6182ba1c1138d32b77fb9f7f37b70034b9f04443a5bea3c188b
eadb36490a7e95f91a4c162acfc3401c3a4f4e5a59251d45243ac8544a665cb951422fa341303f301f0603551d
23041830168014 2318e55671f08eae212142a817720fb817ee93bf300c0603551d130101ff04023000300e0603
551d0f0101ff040403020780300a06082a8648ce3d0403020348003045022 06a552690283860fc94916ebbc92f
```

```
3510abb62a7a4729be57c2bde9fadf417e71022100c2385d82cfb33a357d5402f3e20fb271d0145b72b38a4a2b
4a6ebc6e14dd83b5
```

14.2 Expedited-standard phase with Reader Certificate

```
[Reader] Reader stored reader long term public key:
0418a9f3120accfbd9cc5531018815eb78b97123f8769c389c1cf011d70a64f1f0a4fdee0431afc2fb1d677c5021ed8d1c959a667f4ef15b5a4f
1758fa5f165249
[Reader] Reader stored reader long term private key:
d67eccba434e0735f9247ee2b4ccf531768eed8b8a47a56d40533d6bc99c6ddd
[Reader] Reader stored reader group identifier: 00112233445566778899aabbccddeeff
[Reader] Reader stored reader group sub identifier: ffeeddccbbaa99887766554433221100
[Reader] Reader stored device public key:
04ed1c8b8eb7e44c2842db98730717c75cc94c96ab9ae60f079879e756980b4003b38fb449203f7237cb9f81077b8ac49c75c8115ed408312222
eab61e18feca17
[Reader] Reader stored device public key:
0488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a00973d21
ed758c1471a748
[Reader] Reader stored compressed reader certificate:
308196040200003081 8542000418a9f3120accfbd9cc5531018815eb78b97123f8769c389c1cf011d70a64f1f0a4fdee0431afc2fb1d677c50
21ed8d1c959a667f4ef15b5a4f1758fa5f1652498649003046022100ce9b542d03aff827b09a3530048935 61574331d1c27eb195f933006590d8
848e022100daa125d7221a8b70a32bcf8a36704343cb49f239605c4dcef04235e5c67c5c03
[Reader] using expedited transaction AID: a000000909acce5501
[Reader] starting transaction over NFC transport
[Reader] user device detected
[Reader] generate select command with AID: a000000909acce5501
[User Device] Reader connected
[User Device] received C-APDU >>00a4040009a000000909acce550100
[User Device] select command
[Reader] received R-APDU <<6f158409a000000909acce5501a508800200005c0201009000
[Reader] received tag A5 with content 800200005c020100
[Reader] access protocol version(s) supported by the user device: 0100
[Reader] reader ephemeral public key:
049696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e75e13ebc6d55743ba6a6ffc4ed37a55515a9346fdae311f60be
30421fa6dc61c5
[Reader] reader ephemeral private key: 3c0f74114cd2a021e8066efbaa31dbb97ef0054272192606fd96633a04f66214
[Reader] generate auth0 command
[Reader] transaction code: 0x01
[Reader] fast transaction requested: False
[Reader] transaction identifier: 4165a83667ad0af5ab115247424822e0
[Reader] selected protocol version: 0100
[User Device] received C-APDU
>>8080000081410100420101 5c0201008741049696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e75e13ebc6d55743
ba6a6ffc4ed37a55515a9346fdae311f60be30421fa6dc61c54c104165a83667ad0af5ab115247424822e04d2000112233445566778899aabbcc
ddeeffffeeddccbbaa9988776655443322110000
[User Device] auth0 command
[User Device] Reader selected protocol version: 0100
[User Device] received reader_group_identifier: 00112233445566778899aabbccddeeff
[User Device] received reader_group_sub_identifier: ffeeddccbbaa99887766554433221100
[User Device] received reader ephemeral public key:
049696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e75e13ebc6d55743ba6a6ffc4ed37a55515a9346fdae311f60be
30421fa6dc61c5
[User Device] generate ephemeral private key: 5dbe110969e429e3fbdddef0622ddc1e25b2e74451433dfb2f3d99c0bc46d65b
[User Device] generate ephemeral public key:
04f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d540aee763c64b211a0e825d45bfa28569231c44de0172e4dd1f060a778859c84f772c1
beb543e016e498
[User Device] flag value: 0001
[User Device] found Access Credential with matching reader_group_identifier in user device storage:
00112233445566778899aabbccddeeff
[User Device] expedited standard transaction requested
[Reader] received R-APDU
<<864104f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d540aee763c64b211a0e825d45bfa28569231c44de0172e4dd1f060a778859c84
f772c1beb543e016e4989000
[Reader] received Access Credential ephemeral public key:
04f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d540aee763c64b211a0e825d45bfa28569231c44de0172e4dd1f060a778859c84f772c1
beb543e016e498
[Reader] standard transaction requested
[Reader] to be signed:
4d200011223344556677889 9aabbccddeeffffeeddccbbaa9988776655443322110086 20f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d
540aee763c64b211a0e887209696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74c104165a83667ad0af5ab115247
424822e09304415d9569
[Reader] signing with key: d67eccba434e0735f9247ee2b4ccf531768eed8b8a47a56d40533d6bc99c6ddd
[Reader] reader signature:
6bc01dd9420af5b884d6cf4a3980520a7541971ed342dbfe41e4eb3c429405cd6751dda65c1e033648838d2540b0bb5710fee50dcb4e959a915f
f86123c1ee2a
[User Device] received C-APDU
>>80810000e24101019e406bc01dd9420af5b884d6cf4a3980520a7541971ed342dbfe41e4eb3c429405cd6751dda65c1e033648838d2540b0bb
5710fee50dcb4e959a915ff86123c1ee2a9082009930819604020000308 18f8542000418a9f3120accfbd9cc5531018815eb78b97123f8 769c38
9c1cf011d70a64f1f0a4fdee0431afc2fb1d677c5021ed8d1c959a667f4ef15b5a4f1758fa5f1652498649003046022100ce9b542d03aff827b0
9a3530048935 61574331d1c27eb195f933006590d8848e022100daa125d7221a8b70a32bcf8a36704343cb49f239605c4dcef04235e5c67c5c03
00
```

```
[User Device] auth1 command
[User Device] uncompressing reader certificate
[User Device] issuer_public_key:
04b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60fd8675c7b3cca0e0070dbb999d9d11f67b4517247452ec931ee
f51f047194172a
[User Device] aki extension: 301680149b1ab250afb3920ccd176d78ee35fdda36395e5b
[User Device] serialNumber not present --> use default value
[User Device] issuer not present --> use default value
[User Device] notBefore not present --> use default value
[User Device] notAfter not present --> use default value
[User Device] subject not present --> use default value
[User Device] uncompressed certificate:
308201533081f9a003020102020101300a06082a8648ce3d0403023011310f300d06035504030c06697373756572301e170d3230303130313030
303030305a170d343930313031303030303030305a30123110300e06035504030c077375626a656374305930130 6072a8648ce3d020106082a8648
ce3d0301070342000418a9f3120accfbd9cc5531018815eb78b97123f8769c389c1cf011d70a64f1f0a4fdee0431afc2fb1d677c5021ed8d1c95
9a667f4ef15b5a4f1758fa5f165249a341303f301f0603551d230418301680149b1ab250afb3920ccd176d78ee35fdda36395e5b300c0603551d
130101ff04023000300e0603551d0f0101ff040403020780300a06082a8648ce3d040302034900304602210 0ce9b542d03aff827b09a35300489
3561574331d1c27eb195f933006590d8848e022100daa125d7221a8b70a32bcf8a36704343cb49f239605c4dcef04235e5c67c5c03
[User Device] reader certificate verified successfully
[User Device] to verify:
4d2000112233445566778899aabbccddeeffffeeddccbbaa998877665544332211008620f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d
540aee763c64b211a0e887209696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74c104165a83667ad0af5ab115247
424822e09304415d9569
[User Device] verifying with key:
0418a9f3120accfbd9cc5531018815eb78b97123f8769c389c1cf011d70a64f1f0a4fdee0431afc2fb1d677c5021ed8d1c959a667f4ef15b5a4f
1758fa5f165249
[User Device] ECDH with ephemeral private key: 5dbe110969e429e3fbdddef0622ddc1e25b2e74451433dfb2f3d99c0bc46d65b
[User Device] ECDH output (Zab): 817ee6f93a748143343375eb0d68b0085331ae80c826077008cee779c7460553
[User Device] SHA256 input:
817ee6f93a748143343375eb0d68b0085331ae80c826077008cee779c74605530000000 14165a83667ad0af5ab115247424822e0
[User Device] kdh 5960718f4a8eb8e28a6bb84fd0109bed0f242dfaa5492141b479c980bc529a5d
[User Device] key derivation with key: 5960718f4a8eb8e28a6bb84fd0109bed0f242dfaa5492141b479c980bc529a5d
[User Device] salt:
18a9f3120accfbd9cc5531018815eb78b97123f8769c389c1cf011d70a64f1f0566f6c6174696c652a2a2a2a0011223344556677889 9aabbccdd
eeffffeeddccbbaa998877665544332211005e5c0201009696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74165a8
3667ad0af5ab115247424822e00001a508800200005c020100
[User Device] info: f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d540aee763c64b211a0e8
[User Device] ExpeditedSKReader: e82e79bb4407ba7c21d2a31af3254338dac18943202f7d27929460b51557e0ac
[User Device] ExpeditedSKDevice: 5f7b39c4e47d2f7b2b0c825551e36169b9b0dac35f814d3b70e9fce1cfd00aa0
[User Device] StepUpSK: f15e760a567cc1ce5366ea02be647eea5ebe33a8cde3dcdfccae0f55eb5b4bad
[User Device] BleSK: 699014b9c3957d255d0b46eea8a7e9961ac7844588f76148ea72fb23f4dce01a
[User Device] URSK: f72b485f8d9c1c271a85f0724de1d7938fbcca263c5051ee95c9ebcefcd40b51
[User Device] to be signed:
4d2000112233445566778899aabbccddeeffffeeddccbbaa998877665544332211008620f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d
540aee763c64b211a0e887209696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74c104165a83667ad0af5ab11 5247
424822e093044e887b4c
[User Device] signing with key: 5b0e4716bfb90700984963a32e3ce1721f1bd39b5b8bd952b6423a78fdeedec1
[User Device] signature:
9e40cde387f04de500dd5aaa96ef8b3f132a81f3da396f5da779e90f38f7f5b50398724735372e323c6ec690d56d85507ed7b154d17f0d54fa90
b5aecea8cc91cda7
[User Device] signaling: 5e02003f
[User Device] to be wrapped:
5a410488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a0097
3d21ed758c1471a7489e40cde387f04de500dd5aaa96ef8b3f132a81f3da396f5da779e90f38f7f5b50398724735372e323c6ec690d56d8 5507e
d7b154d17f0d54fa90b5aecea8cc91cda75e02003f
[User Device] iv: 000000000000000100000001
[User Device] key derivation with key: 5960718f4a8eb8e28a6bb84fd0109bed0f242dfaa5492141b479c980bc529a5d
[User Device] salt:
18a9f3120accfbd9cc5531018815eb78b97123f8769c389c1cf011d70a64f1f050657273697374656e742a2a0011223344556677889 9aabbccdd
eeffffeeddccbbaa998877665544332211005e5c0201009696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74165a8
3667ad0af5ab115247424822e00001a508800200005c02010088f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a230
[User Device] info: f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d540aee763c64b211a0e8
[User Device] Kpersistent: 05f4b4cd1cf0f77b7bcd710fdd5019e54500c66e6fee47834c4fc26eebb31de7
[Reader] received R-APDU
<<112b14e6c13667351c2acd26ba9d76cf29cc3f9be1c45e138c89f85309552cf2cac6dc7831256f717b6a9def5ffdff90e9febcbe35ee00cceb
25c685ba88d8b4fd96e402e0ca9997b14ee3489a046c6aabab94586b8ac664c08e1070653c88784253f720d0c2b032394114d033ca52b35ff993
3a7e6bbafa6607097849755a5a047da8f425cb69465279da6d7d1f7ec4ccb6f451f0d149b0f39000
[Reader] ecdh with reader ephemeral private key: 3c0f74114cd2a021e8066efbaa31dbb97ef0054272192606fd96633a04f66214
[Reader] kdh: 5960718f4a8eb8e28a6bb84fd0109bed0f242dfaa5492141b479c980bc529a5d
[Reader] key derivation with key: 5960718f4a8eb8e28a6bb84fd0109bed0f242dfaa5492141b479c980bc529a5d
[Reader] salt:
18a9f3120accfbd9cc5531018815eb78b97123f8769c389c1cf011d70a64f1f0566f6c6174696c652a2a2a2a0011223344556677889 9aabbccdd
eeffffeeddccbbaa998877665544332211005e5c0201009696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74165a8
3667ad0af5ab115247424822e00001a508800200005c020100
[Reader] info: f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d540aee763c64b211a0e8
[Reader] ExpeditedSKReader: e82e79bb4407ba7c21d2a31af3254338dac18943202f7d27929460b51557e0ac
[Reader] ExpeditedSKDevice: 5f7b39c4e47d2f7b2b0c825551e36169b9b0dac35f814d3b70e9fce1cfd00aa0
[Reader] StepUpSK: f15e760a567cc1ce5366ea02be647eea5ebe33a8cde3dcdfccae0f55eb5b4bad
[Reader] BleSK: 699014b9c3957d255d0b46eea8a7e9961ac7844588f76148ea72fb23f4dce01a
[Reader] URSK: f72b485f8d9c1c271a85f0724de1d7938fbcca263c5051ee95c9ebcefcd40b51
[Reader] to unwrap:
112b14e6c13667351c2acd26ba9d76cf29cc3f9be1c45e138c89f85309552cf2cac6dc7831256f717b6a9def5ffdff90e9febcbe35ee00cceb25
```

```
c685ba88d8b4fd96e402e0ca9997b14ee3489a046c6aabab94586b8ac664c08e1070653c88784253f720d0c2b032394114d033ca52b35ff9933a
7e6bbafa6607097849755a5a047da8f425cb69465279da6d7d1f7ec4ccb6f451f0d149b0f3
[Reader] iv: 000000000000000100000001
[Reader] unwrapped:
5a410488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a0097
3d21ed758c1471a7489e40cde387f04de500dd5aaa96ef8b3f132a81f3da396f5da779e90f38f7f5b50398724735372e323c6ec690d56d85507e
d7b154d17f0d54fa90b5aecea8cc91cda75e02003f
[Reader] credential found in reader storage
[Reader] key derivation with key: 5960718f4a8eb8e28a6bb84fd0109bed0f242dfaa5492141b479c980bc529a5d
[Reader] salt:
18a9f3120accfbd9cc5531018815eb78b97123f8769c389c1cf011d70a64f1f050657273697374656e742a2a00112233445566778899aabbccdd
eeffffeeddccbbaa998877665544332211005e5c0201009696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74165a8
3667ad0af5ab115247424822e00001a508800200005c02010088f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a230
[Reader] info: f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d540aee763c64b211a0e8
[Reader] Kpersistent: 05f4b4cd1cf0f77b7bcd710fdd5019e54500c66e6fee47834c4fc26eebb31de7
[Reader] to verify:
4d2000112233445566778899aabbccddeeffffeeddccbbaa998877665544332211008620f88de82325e1de3365f2d34f5c50f5ff84e1a2f9ff9d
540aee763c64b211a0e887209696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74c104165a83667ad0af5ab115247
424822e093044e887b4c
[Reader] verifying with key:
0488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a00973d21
ed758c1471a748
[Reader] signature verification success
[Reader] k_persistent stored on Reader:
[Reader] 05f4b4cd1cf0f77b7bcd710fdd5019e54500c66e6fee47834c4fc26eebb31de7
```

14.3 Expedited-standard phase without Reader Certificate

```
[Reader] Reader stored reader long term public key:
04b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60fd8675c7b3cca0e0070dbb999d9d11f67b4517247452ec931ee
f51f047194172a
[Reader] Reader stored reader long term private key:
7a9e50a19ae385e39b3bf0c75eb5f9c9a5eb4d51f808231835395fd2c1078367
[Reader] Reader stored reader group identifier: 00112233445566778899aabbccddeeff
[Reader] Reader stored reader group sub identifier: ffeeddccbbaa99887766554433221100
[Reader] Reader stored device public key:
04ed1c8b8eb7e44c2842db98730717c75cc94c96ab9ae60f079879e756980b4003b38fb449203f7237cb9f81077b8ac49c75c8115ed408312222
eab61e18feca17
[Reader] Reader stored device public key:
0488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a00973d21
ed758c1471a748
[Reader] using expedited transaction AID: a000000909acce5501
[User Device] Access Credential stored reader group identifier: 00112233445566778899aabbccddeeff
[User Device] Access Credential stored reader long term public key:
04b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60fd8675c7b3cca0e0070dbb999d9d11f67b4517247452ec931ee
f51f047194172a
[User Device] Access Credential stored long term public key:
0488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a00973d21
ed758c1471a748
[User Device] Access Credential stored long term private key:
5b0e4716bfb90700984963a32e3ce1721f1bd39b5b8bd952b6423a78fdeedec1
[User Device] create mock Access Credential
[User Device] Access Credential stored reader group identifier: 00000000000000000000000000000000
[User Device] Access Credential stored reader long term public key:
0442f3f7ac2ffd5100750825d475954fdad1e13e600b00bcb83f0955cdd4ea6ce3043af9c3d73c7de31cf00ba8e9353dbbf696831c57d25a3537
88639c64ab756b
[User Device] Access Credential stored long term public key:
04015731ebbb92ed25321ec99f55c4c00a85007eb0f8032a6e9a4163e2b542d326e244c8d7a42c6302bacfaafe3cc4daf3ff4d96880761c5a714
f1d14664e48aec
[User Device] Access Credential stored long term private key:
59f19aea46cba9ce209804b1c907d9f0dae520ce30152ac00a8307cc1dbe1199
[Reader] starting transaction over NFC transport
[Reader] user device detected
[Reader] generate select command with AID: a000000909acce5501
[User Device] Reader connected
[User Device] received C-APDU >>00a4040009a000000909acce550100
[User Device] select command
[Reader] received R-APDU <<6f158409a000000909acce5501a508800200005c0201009000
[Reader] received tag A5 with content 800200005c020100
[Reader] access protocol version(s) supported by the user device: 0100
[Reader] reader ephemeral public key:
049696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e75e13ebc6d55743ba6a6ffc4ed37a55515a9346fdae311f60be
30421fa6dc61c5
[Reader] reader ephemeral private key: 3c0f74114cd2a021e8066efbaa31dbb97ef0054272192606fd96633a04f66214
[Reader] generate auth0 command
[Reader] transaction code: 0x01
[Reader] fast transaction requested: False
[Reader] transaction identifier: 4165a83667ad0af5ab115247424822e0
[Reader] selected protocol version: 0100
```

```
[User Device] received C-APDU
>>8080000081410100420101 5c0201008741049696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e75e13ebc6d55743
ba6a6ffc4ed37a55515a9346fdae311f60be30421fa6dc61c54c104165a83667ad0af5ab115247424822e04d2000112233445566778899aabbcc
ddeeffffeeddccbbaa998877665544332211 0000
[User Device] auth0 command
[User Device] Reader selected protocol version: 0100
[User Device] received reader_group_identifier: 00112233445566778899aabbccddeeff
[User Device] received reader_group_sub_identifier: ffeeddccbbaa99887766554433221100
[User Device] received reader ephemeral public key:
049696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e75e13ebc6d55743ba6a6ffc4ed37a55515a9346fdae311f60be
30421fa6dc61c5
[User Device] generate ephemeral private key: 70637ee9b40cee568567c69589276888edca7128bb13fb531f9c4f502d8cc65e
[User Device] generate ephemeral public key:
045d75ab60136a2c54ff27b799ee157f3f3329435c0df608de904c920ac29f72bd4274c2edc810a93e240bf5d6394a92c9766b690b2bf5128ae7
0d6e29257ea786
[User Device] flag value: 0001
[User Device] found Access Credential with matching reader_group_identifier in user device storage:
00112233445566778899aabbccddeeff
[User Device] expedited standard transaction requested
[Reader] received R-APDU
<<8641045d75ab60136a2c54ff27b799ee157f3f3329435c0df608de904c920ac29f72bd4274c2edc810a93e240bf5d6394a92c9766b690b2bf5
128ae70d6e29257ea7869000
[Reader] received Access Credential ephemeral public key:
045d75ab60136a2c54ff27b799ee157f3f3329435c0df608de904c920ac29f72bd4274c2edc810a93e240bf5d6394a92c9766b690b2bf5128ae7
0d6e29257ea786
[Reader] standard transaction requested
[Reader] to be signed:
4d20001122334455667788 99aabbccddeeffffeeddccbbaa9988776655443322110086205d75ab60136a2c54ff27b799ee157f3f3329435c0df6
08de904c920ac29f72bd87209696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74c104165a83667ad0af5ab115247
424822e09304415d9569
[Reader] signing with key: 7a9e50a19ae385e39b3bf0c75eb5f9c9a5eb4d51f808231835395fd2c1078367
[Reader] reader signature:
501952e25339019804a7c3a7e4a1f6d993aec8baba7db6c8c20ac450428c2ff390c2188854ef7964927f88040dddf895ef57cce72379ad9688f3
6c5c7de3c294
[User Device] received C-APDU
>>80810000454101019e40501952e25339019804a7c3a7e4a1f6d993aec8baba7db6c8c20ac450428c2ff390c2188854ef7964927f88040dddf8
95ef57cce72379ad9688f36c5c7de3c29400
[User Device] auth1 command
[User Device] to verify:
4d20001122334455667788 99aabbccddeeffffeeddccbbaa9988776655443322110086205d75ab60136a2c54ff27b799ee157f3f3329435c0df6
08de904c920ac29f72bd87209696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74c104165a83667ad0af5ab11524 7
424822e09304415d9569
[User Device] verifying with key:
04b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60fd8675c7b3cca0e0070dbb999d9d11f67b4517247452ec931ee
f51f047194172a
[User Device] ECDH with ephemeral private key: 70637ee9b40cee568567c69589276888edca7128bb13fb531f9c4f502d8cc65e
[User Device] ECDH output (Zab): 9a6797b920d90bf3c7b7ae3f484e1a9c3e6f31da9aec915746b18222836de4f8
[User Device] SHA256 input:
9a6797b920d90bf3c7b7ae3f484e1a9c3e6f31da9aec915746b18222836de4f8000000014165a83667ad0af5ab115247424822e0
[User Device] kdh cd227f01f917ad1dd5252db51c5ad3da1c3028be750a0f4e69c6a5624fca271c
[User Device] key derivation with key: cd227f01f917ad1dd5252db51c5ad3da1c3028be750a0f4e69c6a5624fca271c
[User Device] salt:
b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60f566f6c6174696c652a2a2a2a00112233445566778899aabbccdd
eeffffeeddccbbaa998877665544332211005e5c0201009696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74165a8
3667ad0af5ab115247424822e00001a508800200005c020100
[User Device] info: 5d75ab60136a2c54ff27b799ee157f3f3329435c0df608de904c920ac29f72bd
[User Device] ExpeditedSKReader: f06ab1499102ca96f75cfa6d2e42c7920382d05a22e959325a91eb3aa4d71ce8
[User Device] ExpeditedSKDevice: de82f4f94575da8369febd52dea94ec3dadad6d4406a9efe76098d6a22a8fd5d
[User Device] StepUpSK: b3cdefdb7dae91722efea57ee5f0981b1b3d5e436b406376635bcfd85b562bee
[User Device] BleSK: 8f770f08d0fedea9c441f5f40b1bff1aaad92547729853ceb23a965761d8799f
[User Device] URSK: 9143579775f7b7463e527c9b8f0a581f31ecadff8c82517372666d0bc7a426db
[User Device] to be signed:
4d20001122334455667788 99aabbccddeeffffeeddccbbaa9988776655443322110086205d75ab60136a2c54ff27b799ee157f3f3329435c0df6
08de904c920ac29f72bd87209696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74c104165a83667ad0af5ab11 5247
424822e093044e887b4c
[User Device] signing with key: 5b0e4716bfb90700984963a32e3ce1721f1bd39b5b8bd952b6423a78fdeedec1
[User Device] signature:
9e402f57a5cb8a88c5a300fadb858d17298ed6f9dc01f9abc65e4b4089439868b8d24e93f1e54ca1df0703a76974a847ebafb42a7e90dccc3aae
d788251d155a63e0
[User Device] signaling: 5e02003f
[User Device] to be wrapped:
5a410488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a0097
3d21ed758c1471a7489e402f57a5cb8a88c5a300fadb858d17298ed6f9dc01f9abc65e4b4089439868b8d24e93f1e54ca1df0703a76974a 847eb
afb42a7e90dccc3aaed788251d155a63e05e02003f
[User Device] iv: 000000000000000100000001
[User Device] key derivation with key: cd227f01f917ad1dd5252db51c5ad3da1c3028be750a0f4e69c6a5624fca271c
[User Device] salt:
b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60f50657273697374656e742a2a00112233445566778899aabbccdd
eeffffeeddccbbaa998877665544332211005e5c0201009696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74165a8
3667ad0af5ab115247424822e00001a508800200005c02010088f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a230
[User Device] info: 5d75ab60136a2c54ff27b799ee157f3f3329435c0df608de904c920ac29f72bd
[User Device] Kpersistent: dd309b1738bcec549f4d6e73c6d15bf595f783d729c8ac0fa7a76ec6c8821a2d
```

```
[Reader] received R-APDU
<<caae4715cb099959b6354df09a754bdeb31689e27be440d0c2cfe8d4e5b5d99ba367801c0f4f46485a160840f4e51b42d5b5e420157d64188a
f6d89921ce5fa482f7e51725ba7568e5976cf6e9443fa57b32fd76a6a1b1b3190bd2aa0ee946f48c65dc8f3dc24c652fb9cab1381a68f0737a77
c5e2b1cfbd9884041049d3e37b7126a2d74d7af03a322fbac65d627ef576a8d83e1a887b5be79000
[Reader] ecdh with reader ephemeral private key: 3c0f74114cd2a021e8066efbaa31dbb97ef0054272192606fd96633a04f66214
[Reader] kdh: cd227f01f917ad1dd5252db51c5ad3da1c3028be750a0f4e69c6a5624fca271c
[Reader] key derivation with key: cd227f01f917ad1dd5252db51c5ad3da1c3028be750a0f4e69c6a5624fca271c
[Reader] salt:
b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60f566f6c6174696c652a2a2a2a00112233445566778899aabbccdd
eeffffeeddccbbaa99887766554433221l005e5c0201009696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74165a8
3667ad0af5ab115247424822e00001a508800200005c020100
[Reader] info: 5d75ab60136a2c54ff27b799ee157f3f3329435c0df608de904c920ac29f72bd
[Reader] ExpeditedSKReader: f06ab1499102ca96f75cfa6d2e42c7920382d05a22e959325a91eb3aa4d71ce8
[Reader] ExpeditedSKDevice: de82f4f94575da8369febd52dea94ec3dadad6d4406a9efe76098d6a22a8fd5d
[Reader] StepUpSK: b3cdefdb7dae91722efea57ee5f0981b1b3d5e436b406376635bcfd85b562bee
[Reader] BleSK: 8f770f08d0fedea9c441f5f40b1bff1aaad92547729853ceb23a965761d8799f
[Reader] URSK: 9143579775f7b7463e527c9b8f0a581f31ecadff8c82517372666d0bc7a426db
[Reader] to unwrap:
caae4715cb099959b6354df09a754bdeb31689e27be440d0c2cfe8d4e5b5d99ba367801c0f4f46485a160840f4e51b42d5b5e420157d64188af6
d89921ce5fa482f7e51725ba7568e5976cf6e9443fa57b32fd76a6a1b1b3190bd2aa0ee946f48c65dc8f3dc24c652fb9cab1381a68f0737a77c5
e2b1cfbd9884041049d3e37b7126a2d74d7af03a322fbac65d627ef576a8d83e1a887b5be7
[Reader] iv: 000000000000000100000001
[Reader] unwrapped:
5a410488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a0097
3d21ed758c1471a7489e402f57a5cb8a88c5a300fadb858d17298ed6f9dc01f9abc65e4b4089439868b8d24e93f1e54ca1df0703a76974a847eb
afb42a7e90dccc3aaed788251d155a63e05e02003f
[Reader] credential found in reader storage
[Reader] key derivation with key: cd227f01f917ad1dd5252db51c5ad3da1c3028be750a0f4e69c6a5624fca271c
[Reader] salt:
b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60f506572736973746566e742a2a00112233445566778899aabbccdd
eeffffeeddccbbaa99887766554433221l005e5c0201009696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74165a8
3667ad0af5ab115247424822e00001a508800200005c02010088f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a230
[Reader] info: 5d75ab60136a2c54ff27b799ee157f3f3329435c0df608de904c920ac29f72bd
[Reader] Kpersistent: dd309b1738bcec549f4d6e73c6d15bf595f783d729c8ac0fa7a76ec6c8821a2d
[Reader] to verify:
4d2000112233445566778899aabbccddeeffffeeddccbbaa998877665544332211008620 5d75ab60136a2c54ff27b799ee157f3f3329435c0df6
08de904c920ac29f72bd87209696afe33de58b7d3253d1cba86d14147c16d455e8a27373b38d454af21b70e74c104165a83667ad0af5ab115247
424822e093044e887b4c
[Reader] verifying with key:
0488f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a23075dbdcf67d15bda429db38706a2f15ba90a2ac3c6a00973d21
ed758c1471a748
[Reader] signature verification success
[Reader] k_persistent stored on Reader:
[Reader] dd309b1738bcec549f4d6e73c6d15bf595f783d729c8ac0fa7a76ec6c8821a2d
```

14.4 Expedited-fast phase without Reader Certificate

```
[Reader] starting transaction over NFC transport
[Reader] user device detected
[Reader] generate select command with AID: a000000909acce5501
[User Device] Reader connected
[User Device] received C-APDU >>00a4040009a000000909acce550100
[User Device] select command
[Reader] received R-APDU <<6f158409a000000909acce5501a508800200005c0201009000
[Reader] received tag A5 with content 800200005c020100
[Reader] access protocol version(s) supported by the user device: 0100
[Reader] reader ephemeral public key:
04de8639f30ff8c502559db84059dbc7fde720044a7ed8717eddf0481315313ed32f559a58ccad407d2c5d4f385f6add3587c8f05e87521b1810
66125d2d1a39d8
[Reader] reader ephemeral private key: a1292f46c8dc580999be17b6c747e5a1284353fc80a7ffb7914a2936633455d3
[Reader] generate auth0 command
[Reader] transaction code: 0x01
[Reader] fast transaction requested: True
[Reader] transaction identifier: 2701e4fe10d21e15b216c550b0c5ee68
[Reader] selected protocol version: 0100
[User Device] received C-APDU
>>8080000081410101420101 5c020100874104de8639f30ff8c502559db84059dbc7fde720044a7ed8717eddf0481315313ed32f559a58ccad40
7d2c5d4f385f6add3587c8f05e87521b181066125d2d1a39d84c102701e4fe10d21e15b216c550b0c5ee684d2000112233445566778899aabbcc
ddeeffffeeddccbbaa998877665544332211 0000
[User Device] auth0 command
[User Device] Reader selected protocol version: 0100
[User Device] received reader_group_identifier: 00112233445566778899aabbccddeeff
[User Device] received reader_group_sub_identifier: ffeeddccbbaa99887766554433221100
[User Device] received reader ephemeral public key:
04de8639f30ff8c502559db84059dbc7fde720044a7ed8717eddf0481315313ed32f559a58ccad407d2c5d4f385f6add3587c8f05e87521b1810
66125d2d1a39d8
[User Device] generate ephemeral private key: 8188df8c9fe94cab14bd1075bfd1e4f13f24c9146940e3d6f118e54d8b27249e
```

```
[User Device] generate ephemeral public key:
04507806c74a52a8e9b34d0796e4e2382ab6f9d9d7417179fc338429bda1c2fff92852d5c7f5643f1f24e468a6d998effeea81d23c9857d10040
c2ea150abede89
[User Device] flag value: 0101
[User Device] found Access Credential with matching reader_group_identifier in user device storage:
00112233445566778899aabbccddeeff
[User Device] expedited fast transaction requested
[User Device] found matching reader_group_sub_identifier, calculate cryptogram using:
dd309b1738bcec549f4d6e73c6d15bf595f783d729c8ac0fa7a76ec6c8821a2d
[User Device] key derivation with key: dd309b1738bcec549f4d6e73c6d15bf595f783d729c8ac0fa7a76ec6c8821a2d
[User Device] salt:
b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60f566f6c6174696c6546617374001122334455667788 99aabbccdd
eeffffeeddccbbaa9988776655443322110 05e5c020100de8639f30ff8c502559db84059dbc7fde720044a7ed8717eddf0481315313ed32701e4
fe10d21e15b216c550b0c5ee680101a508800200005c02010088f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a230
[User Device] info: 507806c74a52a8e9b34d0796e4e2382ab6f9d9d7417179fc338429bda1c2fff9
[User Device] cryptogramSK: 46b35933b497ead9d72e024b267ce1db9a59ba54fc73d46bda3149a8b047bcaf
[User Device] ExpeditedSKReader: e1010bdbdc2acf8e9ca3a31680439995aca6261500e870eb349b24ab909b1982
[User Device] ExpeditedSKDevice: aa3d35bf0b073b1321404fc49c4d0fd8a31828f13f4d2fa27da3290796807666
[User Device] BleSK: 576603533baa95bbcf91ceee39dfc8b07e1d09b0eefbf2b7d10648cf038e4563
[User Device] URSK: c967a070ea1c609352632cfaca5ed0bd20ee554226163bc27fe0075313d9f8fe
[User Device] generate cryptogram
[User Device] cryptogramSK: 46b35933b497ead9d72e024b267ce1db9a59ba54fc73d46bda3149a8b047bcaf
[User Device] iv: 000000000000000000000000
[User Device] plaintext:
5e02003f911400000000000000000000000000000000000000000000921400000000000000000000000000000000000000000000
[User Device] cryptogram:
ba76234a1e427f9e463106251fb9e9edc5f5812f59fd887d4e57eb0bc544b7cb9d368c4dedadf782d520a91f9666b9091e0973894522c04b142f
6447b596942a
[Reader] received R-APDU
<<864104507806c74a52a8e9b34d0796e4e2382ab6f9d9d7417179fc338429bda1c2fff92852d5c7f5643f1f24e468a6d998effeea81d23c9857
d10040c2ea150abede899d40ba76234a1e427f9e463106251fb9e9edc5f5812f59fd887d4e57eb0bc544b7cb9d368c4dedadf782d520a91f9666
b9091e0973894522c04b142f6447b596942a9000
[Reader] received Access Credential ephemeral public key:
04507806c74a52a8e9b34d0796e4e2382ab6f9d9d7417179fc338429bda1c2fff92852d5c7f5643f1f24e468a6d998effeea81d23c9857d10040
c2ea150abede89
[Reader] fast transaction requested, trying to match received cryptogram:
ba76234a1e427f9e463106251fb9e9edc5f5812f59fd887d4e57eb0bc544b7cb9d368c4dedadf782d520a91f9666b9091e0973894522c04b142f
6447b596942a
[Reader] derive fast transaction keys using Kpersistent:
dd309b1738bcec549f4d6e73c6d15bf595f783d729c8ac0fa7a76ec6c8821a2d
[Reader] key derivation with key: dd309b1738bcec549f4d6e73c6d15bf595f783d729c8ac0fa7a76ec6c8821a2d
[Reader] salt:
b62d9b8f494f2f43a07a7db7e965865d04feeabe4e9c3b8a2f5a544ee2a9c60f566f6c6174696c6546617374001122334455667788 99aabbccdd
eeffffeeddccbbaa9988776655443322110 05e5c020100de8639f30ff8c502559db84059dbc7fde720044a7ed8717eddf0481315313ed32701e4
fe10d21e15b216c550b0c5ee680101a508800200005c02010088f6f8f2f1e35a58879e72d9ea81957e8964c3d3c566eb9d41c83d0d8c63a230
[Reader] info: 507806c74a52a8e9b34d0796e4e2382ab6f9d9d7417179fc338429bda1c2fff9
[Reader] cryptogramSK: 46b35933b497ead9d72e024b267ce1db9a59ba54fc73d46bda3149a8b047bcaf
[Reader] ExpeditedSKReader: e1010bdbdc2acf8e9ca3a31680439995aca6261500e870eb349b24ab909b1982
[Reader] ExpeditedSKDevice: aa3d35bf0b073b1321404fc49c4d0fd8a31828f13f4d2fa27da3290796807666
[Reader] BleSK: 576603533baa95bbcf91ceee39dfc8b07e1d09b0eefbf2b7d10648cf038e4563
[Reader] URSK: c967a070ea1c609352632cfaca5ed0bd20ee554226163bc27fe0075313d9f8fe
[Reader] cryptogram is matching
[Reader] cryptogram matched! decrypted data:
5e02003f911400000000000000000000000000000000000000000000921400000000000000000000000000000000000000000000
```

14.5 Data Elements Examples

Simple access:

```
Data element bytes:

0xA4613100613258200D7BBE4FFE15818CC01FAACFA215C4497929BFD7E67C210A64208DE7741CCE6B61336B70726F706572747934303461 34A2
0001014A6964676F657368657265

Diagnostic notation:

{
  "1": 0,
  "2": h'0d7bbe4ffe15818cc01faacfa215c4497929bfd7e67c210a64208de7741cce6b',
  "3": "property404",
  "4": {
    0: 1,
    1: h'6964676F657368657265'
  }
}
```

```
Annotated diagnostic notation

{
  AccessData_Version : 1,
  AccessData_ID: h'6964676F657368657265'
}
```

Delivery access:

```
Data element bytes:

0xA4613100613258203D91CF21F5739FE005FF4454BA7626592DBFB8ACF6A21DEE8B0A5291AC54F60561337670726F70657274793430343A6672
6F6E745F646F6F726134A400010149706172636 56C3132330281A2000801010381A3001A6687E040011A6687FC600301

Diagnostic notation:
{
  "1": 0,
  "2": h'3d91cf21f5739fe005ff4454ba7626592dbfb8acf6a21dee8b0a5291ac54f605',
  "3": "property404:front_door",
  "4": {
    0: 1,
    1: h'70617263656c313233',
    2: [
      {
        0: 8,
        1: 1
      }
    ],
    3: [
      {
        0: 1720180800,
        1: 1720188000,
        3: 1
      }
    ]
  }
}

Annotated diagnostic notation:

{
  AccessData_Version: 1,
  AccessData_ID: h'70617263656c313233',
  AccessData_AccessRules: [
    {
      AccessRule_Capabilities: Momentary_Unsecure,
      AccessRule_AllowScheduleIds: AccessDataSchedule1
    }
  ],
  AccessData_Schedules: [
    {
      Schedule_StartPeriod: 1720180800,
      Schedule_EndPeriod: 1720188000,
      Schedule_Flags: Time_in_UTC
    }
  ]
}
```

Weekday schedule:

Allow schedule for M-F 8-18:

```
Data element bytes:

0xA461310061325820 72BBDFE56AF210D6A168BCAB6192D97D671C8B0BB5AE11B35E32970AD3570FD2613370736F6D652E6578616D706C652E6F
72676134A4000101466674653A39390281A2000801010381A3001A6593C2800285198CA0181F0201000301

Diagnostic notation:
{
  "1": 0,
  "2": h'72bbdfe56af210d6a168bcab6192d97d671c8b0bb5ae11b35e32970ad3570fd2',
  "3": "some.example.org",
```

```
  "4": {
    0: 1,
    1: h'6674653a3939',
    2: [
      {
        0: 8,
        1: 1
      }
    ],
    3: [
      {
        0: 1704182400,
        2: [
          36000,
          31,
          2,
          1,
          0
        ],
        3: 1
      }
    ]
  }
}

Annotated diagnostic notation:

{
  AccessData_Version: 1,
  AccessData_ID: h'6674653a3939',
  AccessData_AccessRules: [
    {
      AccessRule_Capabilities: Momentary_Unsecure,
      AccessRule_AllowScheduleIds: AccessDataSchedule1
    }
  ],
  AccessData_Schedules: [
    {
      Schedule_StartPeriod: 1704182400,
      Schedule_RecurrenceRule: [
        RecurrenceRule_DurationSeconds: 36000,
        RecurrenceRule_Mask: Monday, Tuesday, Wednesday, Thursday, Friday,
        RecurrenceRule_Pattern: Weekly,
        RecurrenceRule_Interval: 1,
        RecurrenceRule_Ordinal: 0
      ],
      Schedule_Flags: Time_in_UTC
    }
  ]
}
```

14.6 Step-up phase Example

```
SKReader StepUpSK: 616f6575616f6575616f6575616f6575616f6575616f6575616f6575616f6575
SKReader key: d8dcf4959bf4ae5f05318bbd47d793f00bcb1dfaa82efbb32e10933c86148478

SKDevice StepUpSK: 616f6575616f6575616f6575616f6575616f6575616f6575616f6575616f6575
SKDevice key: 6a57227cc56f84760f03cd6c55c4da55d4a85cdc4ef39bb69a4fdf466606b270

DeviceRequest:
a2613163312e30613281a16131d818582ba26131a167616c69726f2d61a268656c656d656e7432f568656c656d656e7434f5613567616c69726f
2d61

DeviceResponse:
a3613163312e30613281a26131a26131a167616c69726f2d6181d8185838a4613101613258200aa260c85ca2f6eca90016720a1d7c7c160baf9c
fa1a5aa4156331b71863b426613368656c656d656e74326134a1000161328443a10126a104478ea23b8fe54e51590133d81859012ea761316331
2e306132675348412d3235366133a167616c69726f2d61a3005820b193e9b1fd40d43aee51f794fb2754f537a12104b743f53ede26d4a74ef604
660158202f6f396adb893a91242c60f3b3a32237c90f543cbbed2bf10398ac228955b7e902582095feb0333d71a311b94921230db1bcd094629c
```

```
01d0fe5e1f2ab6d888b8997ca36134a16134a401022001215820963l3d6c63e24e3372742bfdb1a33ba2c897dcd68ab8c753e4fbd48dca6b7f9a
2258201fb3269edd418857de1b39a4e4a44b92fa484caa722c228288f01d0c03a2c3d6613567616c69726f2d616136a36131c074323032342d30
362d30315431333a33303a30325a6132c074323032342d30362d30315431333a33303a30325a6133c074323032352d30362d30315431333a3330
3a30325a6137f5584007df311fce5e28c83b5b88e6402fae24250c778eec0c58e06283a7d6ab7037e791307aadb8571b1229e18c49932de464a4
dc4f639ad186eb8742099b56a15d17613567616c69726f2d61613300

DeviceRequest diagnostic
{
  "1": "1.0",
  "2": [
    {
      "1": 24( << {
        "1": {
          "aliro-a": {
            "element2": true,
            "element4": true
          }
        },
        "5": "aliro-a"
      } >> )
    }
  ]
}

DeviceResponse diagnostic
{
  "1": "1.0",
  "2": [
    {
      "1": {
        "1": {
          "aliro-a": [
            24( << {
              "1": 1,
              "2": h'0aa260c85ca2f6eca90016720a1d7c7c160baf9cfa1a5aa4156331b71863b426',
              "3": "element2",
              "4": {
                0: 1
              }
            } >> )
          ]
        },
        "2": [
          << {
            1: -7
          } >>,
          {
            4: h'8ea23b8fe54e51'
          },
          << 24( << {
            "1": "1.0",
            "2": "SHA-256",
            "3": {
              "aliro-a": {
                0: h'b193e9b1fd40d43aee51f794fb2754f537a12104b743f53ede26d4a74ef60466',
                1: h'2f6f396adb893a91242c60f3b3a32237c90f543cbbed2bf10398ac228955b7e9',
                2: h'95feb0333d71a311b94921230db1bcd094629c01d0fe5e1f2ab6d888b8997ca3'
              }
            },
            "4": {
              "4": {
                1: 2,
                -1: 1,
                -2: h'96313d6c63e24e3372742bfdb1a33ba2c897dcd68ab8c753e4fbd48dca6b7f9a',
                -3: h'1fb3269edd418857de1b39a4e4a44b92fa484caa722c228288f01d0c03a2c3d6'
              }
            },
            "5": "aliro-a",
            "6": {
              "1": 2024-06-01 13:30:02+00:00,
              "2": 2024-06-01 13:30:02+00:00,
              "3": 2025-06-01 13:30:02+00:00
            },
            "7": true
```

```
          } >> ) >>,

h'07df311fce5e28c83b5b88e6402fae24250c778eec0c58e06283a7d6ab7037e791307aadb8571b1229e18c49932de464a4dc4f639ad186eb87
42099b56a15d17'
        ]
      },
      "5": "aliro-a"
    }
  ],
  "3": 0
}

SessionData request:
a16464617461584c8c8fba6253f17dd60f3f30fcf035195ecca10e706320c72ed41920e59ad72d0002305f08dfd03f785d91eb22cfe475760b72
cba8c427a15c4520de417fc1a8b13c80f4a66ddc19df0e5b5d17

SessionData response:
a164646174615901fc8068130e2312181a60e00fbb74c6f71431b8dd77ec38d0e622568ec417f74575e2dcce0623bcff99c3fc3bf8a90bfdad1f
1263ff016dc43f44c518c472de38ba2f05c2dff4493dd713d0babde2c009c5842d5cb7d348f1d93c1ca3224581a0ffc320631a86e93be1b2d572
846ceb46746e2438fd21179c1587d9f48e941b569682649ceac641c9a7a4dcdcd4f17de66b5c7660341e47d41785d578244f80d1ce00edf7a9b4
b2ef2b2a5850c5f365f6a85d9ebf1fd0a8b9523d2d467ed1fd4dba8a95ebd8e46c1969085150738603bcb645dde6f5ebced559d87582601c2b7d
b09d7a056059796084843b928559f81af2761d7c32dc101723f15f50c543212631590cc371e0010eb48011e370de65e7330b6ea4eb0bbe666e63
f01451681b8e1261c12bca61524ad2daf081c9d21d06f30231eb926dce6e1c52f92c1068cea7a6ca8fe11b5dfd1d6fe48e8c336c2ef11f863f98
dc877625ba8710cb7c860ee7f7202662409254e9c2da0cc2bf00ad585189e301a013342123782cdacec9eb1b17ddbc99eff438c501ddfb6058ef
36db28a44b32b64470f49dbade000fe91d5925a129f89af53a61527702bc60c4667bc6bcbd9294b6bcaed191dd76bf2873260dbe6da3d29fbeb8
83335bf7a1892421207968e1b87987b8419ffa629b8669582ff1cca8f211078f9b87db132ffdacc6068335d72546a679816d49d8a4
```

15 Appendix with performance requirements

The User Device and Reader SHOULD have processing times for the performance tests as defined in the test specification.

16 Appendix with Security Non-Normative guidance

16.1 General concerns

This section provides guidance for general concerns with a Reader or User Device.

16.1.1 Secure boot

Reader and User Device should perform secure boot using best practices. See section 5.5.3.1 in [23] for details.

16.1.2 Software (firmware) update strategy

Reader and User Device should be capable of performing software (firmware) update or download. They should:

- Ensure authenticity and integrity of software (firmware) updates using best practice cryptography.
- Employ automatic software (firmware) update installation methods.
- Check for available updates at least once after initialization and then periodically.
- Enable automatic software (firmware) updates by default.
- Allow an authorized entity to enable, disable, or postpone installation of security updates.

See section 5.5.5 in [23] for details.

Reader and User Device should prevent unauthorized rollback of the software (firmware) to an earlier version with known vulnerabilities. However, they may include an authorized special update process for rolling back the installed software (firmware) to an earlier version in case the newer rolled out software (firmware) does not behave as expected in the field. Rolling back to earlier versions of software (firmware) should only occur when no security functionality is impacted. However, if rolling back results in the introduction of security vulnerabilities then it should be avoided.

16.1.3 Trust anchors

In the context of software updates, trust anchors such as root CA public keys or root public keys to verify software (firmware) updates should not be modifiable beyond provisioning.

The Reader manufacturer should use a hardware mechanism like a write latch to disable changes to the Credential Issuer root certificate or Credential Issuer public key (see section 6.5 for details) except when the device is in provisioning mode.

This best practice applies even more to any public key or certificate used to verify software (firmware) updates for the Reader or its components.

16.1.4 Hardening against attacks

Reader and User Device should prioritize the use of certified libraries for security and cryptographic functionality. Examples could include certification from Common Criteria.

Reader and User Device should make use of isolated processing approaches for security functions using methods such as Trusted Execution Environment, on-chip secure enclave or dedicated Secure Element. See section 5.5.8.1 in [23] for details. For higher risk use cases a dedicated Secure Element should be used. For lower risk use cases, an on-chip secure enclave may be sufficient.

16.1.5 Random number generation

Random number generation is described in section 8.3.1.1.

16.1.6 Traceable identifiers

Data that could be used to track an individual or specific User Device, such as the ID (see section 7.3.2) should have confidentiality protection when in transit.

16.2 Aliro Specific Informative Guidance

This section provides Aliro specific informative guidance for a Reader or User Device.

16.2.1 Mailbox

The mailbox (see section 8.3.1.15) on User Device and Reader should not be accessible by any other means or interfaces than defined in this specification. Protection of data at rest in mailboxes on the User Device or reader is beyond the scope of the Access Control protocol. These protections are provided by the execution environments in which the mailboxes are instantiated (e.g. secure element, operating system). It is recommended that implementers protect data stored in a remote mailbox (e.g. through encryption and integrity protection) to limit any dependency on the security assurance of the mailbox itself.

16.2.2 Revocation list protection

If the Reader maintains its revocation list using a method that does not retain the original Revocation Document(s) containing the revocation list(s), the Reader should store the revocation data in a way that is resistant to tampering (i.e., attempts to modify or clear the revocation list through direct manipulation of the device's non-persistent storage). For example, the revocation data could be stored in memory with integrity protection such as a MAC or authentication tag generated from a storage encryption key used to verify the data.

16.2.3 Long term keys

Long term asymmetric keys *PrivK should be stored using isolated processing approaches (see section 16.1.4). They should not be accessible outside the isolated processing environment.

Long term symmetric keys (Kpersistent) should be generated and stored using isolated processing approaches. They should not be accessible outside the isolated processing environment.

16.2.4 Ephemeral keys for ECKA-DH

Reader and User Device should generate and process their ephemeral private key *ePrivK using isolated processing approaches to protect its confidentiality (see #1 in 5.6.3.3 in [24] for details).

Reader and User Device should generate their ephemeral private key *ePrivK as close to the time of use as possible and destroy it as soon as possible after use (see #1 and #2 in section 5.6.3.3 in [24] for details).

Reader and User Device should perform the ECC Partial Public-Key Validation Routine (see section D.1.1.1 in [25] or section 5.6.2.3.4 in [24] for details) on externally provided elliptic curve public keys *ePubK before use. The User Device should abort the transaction if the validation routine fails for the reader_ePubK in the AUTH0 command. The Reader should abort the transaction if the validation routine fails for the credential_ePubK in the AUTH0 response.

16.2.5 Volatile symmetric keys

Volatile keys (e.g., kdh, ExpeditedSKReader, ExpeditedSKDevice, BleSK, StepUpSK, and URSK) should be derived and processed using isolated processing approaches to protect their confidentiality. Once those keys are no longer in use, they should be destroyed as soon as possible.

16.2.6 Transport of session keys

If session keys are transferred from one isolated processing environment to another, they should be confidentiality protected during transport. An example is the URSK which may need to be transferred to an external UWB chip. For instance, Secure Channel Protocol (see [26]) is a suitable transport layer for communication between the secure element and the UWB chip.

Another example is the StepUpSK in case it needs to be accessed by a different isolated processing environment than the one where it was derived.

17 Appendix on UWB Ranging Hopping Sequence

This section describes the method to generate the hopping sequence.

17.1 Default Hopping Sequence

This method is mandatory and SHALL be supported by User Device and the Reader.

For the *k*-th ranging session with $\boldsymbol{N_{Round}^{k}}$ ranging rounds per block and hopping key $\boldsymbol{HOP\_Key\_RW^{k}}$, the following function is used to generate the hopping round $\boldsymbol{h^{k}}$ index for the first active ranging round for ranging block *i* :

$$h^{k}(i, HOP\_Key\_RW^{k}, N_{Round}^{k}) = \left(\left(\left((i + HOP\_Key\_RW^{k})\&0xFFFF\right)^{2} mod(2^{16} - 15)\right)\left(N_{Round}^{k} - O^{k}\right)\right) >> 16$$

The hopping index for the second active ranging round is given by $f^{k}=h^{k}+O^{k}$.

The initiator SHALL generate and send the hopping key $\boldsymbol{HOP\_Key\_RW^{k}}$ to the responder device during the ranging session setup process (see section 12.1.4).

18 Appendix on Mailbox Data Format

It is the responsibility of the entity writing the mailbox content to ensure data is kept formatted as follow:

Table 18-1 – Mailbox Data Format

Tag	Length	Tag	Length	Description	Field is
0x60	variable			Start of mailbox data	mandatory
		0x81	variable	Index: [OUI_1(3 bytes) | Type(1 byte) | Offset1(2 bytes)] [OUI_2 | Type | Offset2] [OUI_3 | Type | Offset3] …	mandatory
		0x82	variable	Data: data at Offset1 | data at Offset2 | data at Offset3 | …	mandatory

The entity writing the mailbox SHALL comply with the following requirements:

-When no data is present in the mailbox, all bytes contained in the mailbox SHALL be set to 0x00.

-The entries listed in the Index field SHALL use an OUI or CID allocated by IEEE to prevent name collisions.

-The entries listed in the Index field SHALL be sorted by increasing Offset value.

-The first Offset listed in the Index field SHALL have value 0.

-Type is a 1 byte value defined by the vendor, this value can be used to differentiate among multiple data types.

-OffsetX is the position of a particular vendor data relative to the beginning of the Data field.

Note that minimal form of the mailbox data format with 1 OUI, Type and Offset has the following form:

Table 18-2 – Example of minimal Mailbox Data Format

Tag	Length	Value		
0x60	variable	Tag	Length	Value
		0x81	0x06	OUI | Type | Offset
		0x82	Variable	Data

19 Appendix on informative cryptographic summary

This informative annex lists all the cryptographic primitives that are explicitly mentioned in this specification. It is not a complete list of all the cryptographic primitives required to be implemented to support this specification since other primitives may be required by the specifications referenced in this specification. This list of cryptographic primitives is not a normative list, and only intended to be used as reference information.

Curves:

P-256 according to [5].

Ciphers:

AES-128 according to [21]

AES-256 according to [21]

Hashing algorithms:

SHA-256

SHA-1

Signatures:

ECDSA with P-256 according to [5].

Diffie-Hellman:

ECKA-DH with P-256 according to [4]

Cipher modes:

AES-GCM according to [8]

Key derivation:

HKDF with HMAC-SHA-256 according to [10] and [20]

The following sections describe cryptographic operations:

Section 7.2.1 for access document

Section 7.4 for credential verification

Section 8.3.1 for expedited-phase security

Section 8.4.3 step-up security

Section 11.3.1 Bluetooth LE tag generation

Section 11.8 Access control message security

Section 13.1 certificate requirements

20 Appendix on BLE dynamic tag examples

```
Group Resolving Key: f5b165224a58b791df6af1d8303e61cd
Advertising Address: c4:bb:86:c3:27:10 (MSB first representation)
Expiry Timestamp: 0x7a4b8500 (2035-01-07 08:00:00)
Padding bytes: 000000000000
Plaintext to be encrypted: 000000000000c4bb86c327107a4b8500
Ciphertext: 7b7f4a8255799029f0014c3a7726b8df
Dynamic Tag: 7b7f4a82557990 (MSB first representation)

Group Resolving Key: 3c344c4189eb2f1e7bd5d47e446fcec2
Advertising Address: a3:d8:11:73:e5:78 (MSB first representation)
Expiry Timestamp: 0x7a4b8500 (2035-01-07 08:00:00)
Padding bytes: 000000000000
Plaintext to be encrypted: 000000000000a3d81173e5787a4b8500
Ciphertext: ef67e4681a7783103907ed99908cc2eb
Dynamic Tag: ef67e4681a7783 (MSB first representation)

Group Resolving Key: 1bcccea696762e6116c6e9c92d99bf35
Advertising Address: 8c:2e:07:18:e4:7c (MSB first representation)
Expiry Timestamp: 0x7a4b8500 (2035-01-07 08:00:00)
Padding bytes: 000000000000
Plaintext to be encrypted: 0000000000008c2e0718e47c7a4b8500
Ciphertext: d4dd12a45037ba80dd8ee8d136122106
Dynamic Tag: d4dd12a45037ba (MSB first representation)
```

21 Appendix with CDDL definitions

The CDDL for the Access Data Element is

```
; Access Data Element
AccessData = {
    AccessData_Version => uint,
    ? AccessData_ID => bstr .size (1..16),
    ? AccessData_AccessRules => [1*8 AccessRule],
    ? AccessData_Schedules => [1*8 Schedule],
    ? AccessData_ReaderRuleIds => [1*8 uint .size 2],
    ? AccessData_NonAccessExtensions => {+ Vendor_RegisteredID => [+ NonAccessExtension] },
    ? AccessData_AccessExtensions => {+ Vendor_RegisteredID => [+ AccessExtension] }
}

Vendor_RegisteredID = uint .size 3

AccessRule = {
    ? AccessRule_Capabilities => uint .bits AccessRuleCapabilitiesBits,
    ? AccessRule_AllowScheduleIds => uint .bits AccessRuleScheduleIdsBits,
    ? AccessRule_DenyScheduleIds => uint .bits AccessRuleScheduleIdsBits
}

AccessRuleCapabilitiesBits = &(
    Secure                      : 0,
    Unsecure                    : 1,
    Toggle_Secure_or_Unsecure   : 2,
    Momentary_Unsecure          : 3,
    Extended_Momentary_Unsecure : 4,
    Payment_Permission          : 5
) / (6..15) ; RFU

AccessRuleScheduleIdsBits = &(
    AccessDataSchedule1 : 0,
    AccessDataSchedule2 : 1,
    AccessDataSchedule3 : 2,
    AccessDataSchedule4 : 3,
    AccessDataSchedule5 : 4,
    AccessDataSchedule6 : 5,
    AccessDataSchedule7 : 6,
    AccessDataSchedule8 : 7,
)

Schedule = {
    Schedule_StartPeriod => uint .size 4,
    ? Schedule_EndPeriod => uint .size 4,
    ? Schedule_RecurrenceRule => RecurrenceRule,
    Schedule_Flags => uint .bits Schedule_FlagsBits
}

RecurrenceRule = [
    RecurrenceRule_DurationSeconds : uint .size 4
    RecurrenceRule_Mask : uint .bits RecurrenceRuleMaskBits / 0,
    RecurrenceRule_Pattern : RecurrenceRulePatternType,
    RecurrenceRule_Interval : uint .size 1,
    RecurrenceRule_Ordinal : RecurrenceRuleOrdinalValue,
]

Schedule_FlagsBits = &(
    Time_in_UTC: 0,
) / (1..7) ; RFU


RecurrenceRulePatternType = &(
    Daily               : 1,
    Weekly              : 2,
    MonthlyByWeekDay    : 3,
    MonthlyByDate       : 4,
    YearlyByWeekDay     : 5,
    YearlyByDate        : 6,
    YearlyByWeek        : 7,
    YearlyByMonthWeek   : 8
```

```
)

RecurrenceRuleOrdinalValue = &(
    RecurrenceRuleOrdinal_Daily             : 0,
    RecurrenceRuleOrdinal_Weekly            : 0,
    RecurrenceRuleOrdinal_MonthlyByWeekday  : (-5..-1) / (1..5),
    RecurrenceRuleOrdinal_MonthlyByDate     : ((-31..-1) / (1..31)) / 0,
    RecurrenceRuleOrdinal_YearlyByWeekday   : (-5..-1) / (1..5),
    RecurrenceRuleOrdinal_YearlyByDate      : (-31..-1) / (1..31),
    RecurrenceRuleOrdinal_YearlyByWeek      : (-53..-1) / (1..53),
    RecurrenceRuleOrdinal_YearlyByMonthWeek : (-5..-1) / (1..5)
)

RecurrenceRuleMaskBits = &(
    RecurrenceRuleMask_Weekly           : RecurrenceRuleMaskBits_Weekdays,
    RecurrenceRuleMask_MonthlyByWeekDay : RecurrenceRuleMaskBits_Weekdays,
    RecurrenceRuleMask_MonthlyByDate    : RecurrenceRuleMaskBits_Dates /
RecurrenceRuleMaskBits_Weekdays,
    RecurrenceRuleMask_YearlyByWeekDay  : RecurrenceRuleMaskBits_Yearly,
    RecurrenceRuleMask_YearlyByDate     : RecurrenceRuleMaskBits_Yearly,
    RecurrenceRuleMask_YearlyByWeek     : RecurrenceRuleMaskBits_Yearly,
    RecurrenceRuleMask_YearlyByMonthWeek : RecurrenceRuleMaskBits_Yearly
)

RecurrenceRuleMaskBits_Weekdays = &(
    Monday    : 0,
    Tuesday   : 1,
    Wednesday : 2,
    Thursday  : 3,
    Friday    : 4,
    Saturday  : 5,
    Sunday    : 6
)

RecurrenceRuleMaskBits_Yearly = &(
    Monday    : 0,
    Tuesday   : 1,
    Wednesday : 2,
    Thursday  : 3,
    Friday    : 4,
    Saturday  : 5,
    Sunday    : 6,
    January   : 7,
    February  : 8,
    March     : 9,
    April     : 10,
    May       : 11,
    June      : 12,
    July      : 13,
    August    : 14,
    September : 15,
    October   : 16,
    November  : 17,
    December  : 18
)

RecurrenceRuleMaskBits_Dates = &(
    day1     : 0,
    day2     : 1,
    day3     : 2,
    day4     : 3,
    day5     : 4,
    day6     : 5,
    day7     : 6,
    day8     : 7,
    day9     : 8,
    day10    : 9,
    day11    : 10,
    day12    : 11,
    day13    : 12,
    day14    : 13,
    day15    : 14,
```

```
    day16    : 15,
    day17    : 16,
    day18    : 17,
    day19    : 18,
    day20    : 19,
    day21    : 20,
    day22    : 21,
    day23    : 22,
    day24    : 23,
    day25    : 24,
    day26    : 25,
    day27    : 26,
    day28    : 27,
    day29    : 28,
    day30    : 29,
    day31    : 30,
)

NonAccessExtension = [
    Vendor_ExtensionID : uint,
    Version : uint,
    Data : bstr
]

AccessExtension = [
    Criticality : uint .bits Criticality_Bits ,
    Vendor_ExtensionID : uint,
    Version : uint,
    Data : bstr
]

Criticality_Bits = &(
    Critical        : 0
) / (1..7) ; RFU

; AccessData Labels
AccessData_Version = 0
AccessData_ID = 1
AccessData_AccessRules = 2
AccessData_Schedules = 3
AccessData_ReaderRuleIds = 4
AccessData_NonAccessExtensions = 5
AccessData_AccessExtensions = 6

; AccessRule Labels
AccessRule_Capabilities = 0
AccessRule_AllowScheduleIds = 1
AccessRule_DenyScheduleIds = 2

; Schedule Labels
Schedule_StartPeriod = 0
Schedule_EndPeriod = 1
Schedule_RecurrenceRule = 2
Schedule_Flags = 3
```

The CDDL for the Revocation Data Element is:

```
RevocationData = {
    RevocationData_Version => uint,
    RevocationData_ChangeMode => ChangeMode,
    ? RevocationData_Entries => [+RevocationEntry],
    ? RevocationData_EntriesRemove => [+RevocationEntry],
    ? RevocationData_Extensions => {+ Vendor_RegisteredID => [+ Extension] }
}

Vendor_RegisteredID = uint .size 3

ChangeMode = &(
    Overwrite : 0,
    Update    : 1
)
```

```
RevocationEntry = {
    ? RevocationEntry_PublicKeyHash => bstr,
    ? RevocationEntry_ID => bstr .size (1..16),
    ? RevocationEntry_ExpiryTime => uint .size 4
}

Extension = [
    Vendor_ExtensionID : uint,
    Version : uint,
    Data : bstr
]

RevocationData_Version = 0
RevocationData_ChangeMode = 1
RevocationData_Entries = 2
RevocationData_EntriesRemove = 3
RevocationData_Extensions = 4

RevocationEntry_PublicKeyHash = 0
RevocationEntry_ID = 1
RevocationEntry_ExpiryTime = 2
```