Find My Network Accessory Specification

Draft v1

R3

Developer

Contents

1. Introduction

NOTICE OF PROPRIETARY PROPERTY: THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS THE PROPRIETARY PROPERTY OF APPLE INC. THE POSSESSOR AGREES TO THE FOLLOWING: (I) TO MAINTAIN THIS DOCUMENT IN CONFIDENCE, (II) NOT TO REPRODUCE OR COPY IT, (III) NOT TO REVEAL OR PUBLISH IT IN WHOLE OR IN PART, (IV) ALL RIGHTS RESERVED.

ACCESS TO THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS GOVERNED BY THE TERMS OF THE MFi LICENSE AGREEMENT AND FIND MY NETWORK SUPPLEMENT. ALL OTHER USE SHALL BE AT APPLE'S SOLE DISCRETION.

1.1. Requirements, recommendations, and permissions

This specification contains statements that are incorporated by reference into legal agreements between Apple and its Licensees. The use of the words *must, must not, required, shall, shall not, should, should not, recommended, not recommended, may, optional,* and *deprecated* in a statement have the following meanings:

- *Must, shall*, or *required* means the statement is an absolute requirement.
- *Must not, shall not,* or *prohibited* means the statement is an absolute prohibition.
- *Should* or *recommended* means the full implications must be understood before choosing a different course.
- *Should not* or *not recommended* means the full implications must be understood before choosing this course.
- *May* or *optional* means the statement is truly optional, and its presence or absence cannot be assumed.
- *Deprecated* means the statement is provided for historical purposes only and is equivalent to "must not."

The absence of requirements, recommendations, or permissions for a specific accessory design in this specification must not be interpreted as implied approval of that design. Licensee is strongly encouraged to ask Apple for feedback on accessory designs that are not explicitly mentioned in this specification.

1.2. Terminology

Throughout this document, these terms have specific meanings:

- The term *Apple device* is used to refer to an iPhone, iPad, iPod touch, Apple Watch, or Mac (running iOS, iPadOS, watchOS, or macOS).
- The term *accessory* is used to refer to any product intended to interface with a device through the means described in this specification.
- The term *Apple Account* is an authentication method that Apple uses for iPhone, iPad, Mac, and other Apple devices and services. When an Apple Account is used to log in to an Apple device, the device will automatically use the settings associated with the Apple Account.
- α ll β denotes concatenation of the values α and β.

2. Core Concepts

2.1. Overview

The *Find My Network Accessory Specification* defines how an accessory communicates with Apple devices to help owners locate their accessories privately and securely by using the Find My network.

2.2. Find My app

The Find My app is where you locate your Apple devices, share your location with friends and family, and locate Find My network-enabled accessories. The app displays the location of findable items and includes additional features to protect your devices, such as playing sound and using Lost Mode. See the Find My webpage for more details.

2.3. Transport

The Find My network accessory protocol uses Bluetooth Low Energy (LE) as the primary transport to interact with Apple devices.

2.4. Operation

The accessory and the owner Apple device generate a cryptographic key pair after Find My network pairing. The owner Apple device has access to both the private and the public key, and the accessory has the public key.

An accessory subsequently broadcasts a rotating key (derived from the public key) as a low energy Bluetooth beacon. This beacon can be picked up by nearby Apple devices (see Find My network). The Apple devices publish the key received in the Bluetooth beacon, along with its own location encrypted using that same key, to Apple servers. Because the private key is stored only on the owner device, the location information is accessible only to the device owner. The data stored in Apple servers is end-to-end encrypted, and Apple does not have access to any location information.

2.5. Roles

2.5.1. Owner device

When an accessory is paired with an Apple device through the Find My app, the accessory is associated with the Apple Account on that device. This device and all other Apple devices signed in with the same Apple Account are treated as owner devices.

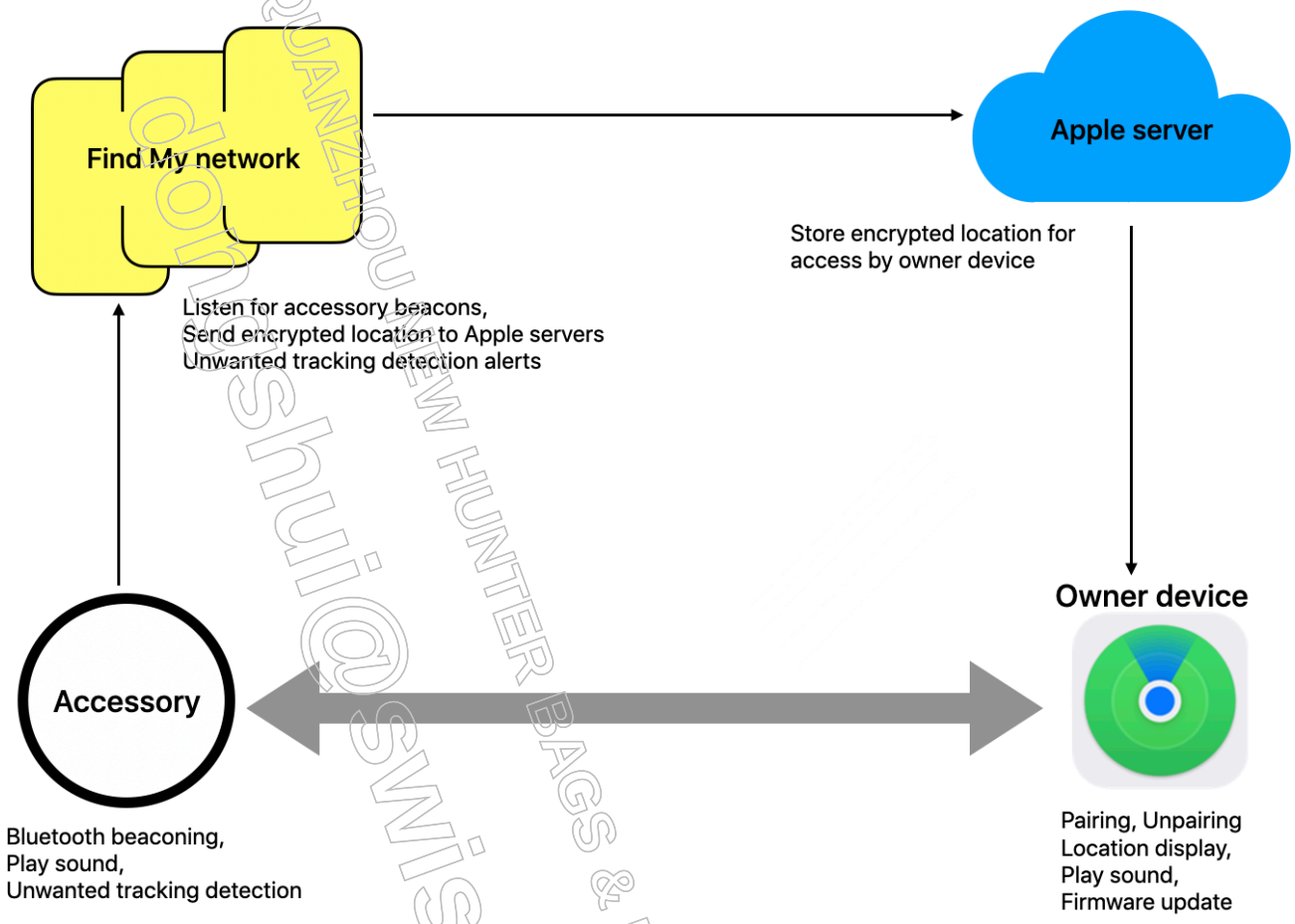

Figure 2-1: Different roles in the Find My network

The Find My app on an owner device can be used to locate accessories. An owner device is required for actions such as unpairing the device, firmware update, locate, and so on.

2.5.2. Accessory

An *accessory* is the device that implements the Find My network accessory protocol and can be located using the Apple Find My network and servers. The accessory is paired with the Apple Account in use on the owner device.

2.5.3. Find My network

The Find My network provides a mechanism to locate accessories by using the vast network of Apple devices that have Find My enabled. When an accessory is detected by a nearby Apple device, the device publishes its own encrypted location as the approximate location of the detected accessory. Reports from more than one Apple device can provide a more precise location. Any Apple device that participates in the Find My network is called a *Finder device*. Participation in the Find My network is a user choice that can be reviewed or changed anytime in Settings.

A *non-owner device* refers to a device that may connect to the accessory but is not an owner device. (For example, a device might connect in response to a UT alert; see Unwanted tracking detection.)

2.5.4. Apple server

Apple server receives encrypted location data from Finder devices and temporarily stores it. Only the owner devices can decrypt and read raw locations from the encrypted data. Apple cannot read this information.

2.6. Features

2.6.1. Unwanted tracking detection

Unwanted tracking detection (UT) notifies the user of the presence of an unrecognized accessory that may be traveling with them over time and allows them to take various actions, including playing a sound on the accessory if it's in Bluetooth LE range.

2.6.2. Lost mode

An owner can use the Find My app to place their accessory in Lost Mode. They can set a phone number and select a message from a predefined list.

When someone finds someone else's lost accessory, they can discover the details set by the owner by using NFC or Bluetooth LE to help the owner recover the lost item. See Serial number lookup for details.

2.6.3. Play sound

The Play sound feature allows users to play sound from their Apple device to locate the accessory. This action may be initiated from an owner or non-owner device.

Users can play a sound from the Find My app on an owner device. The Apple device creates a Bluetooth LE connection or uses an existing connection to the accessory and uses the Play sound—owner control point to initiate the action.

Users can play a sound from a non-owner device when a UT alert appears on that device. That device creates a Bluetooth LE connection and uses the Play sound—non-owner control point to initiate the action.

2.7. States

Accessory operations can be described using a state machine with the states listed in this section and transition between them based on interactions with an owner device.

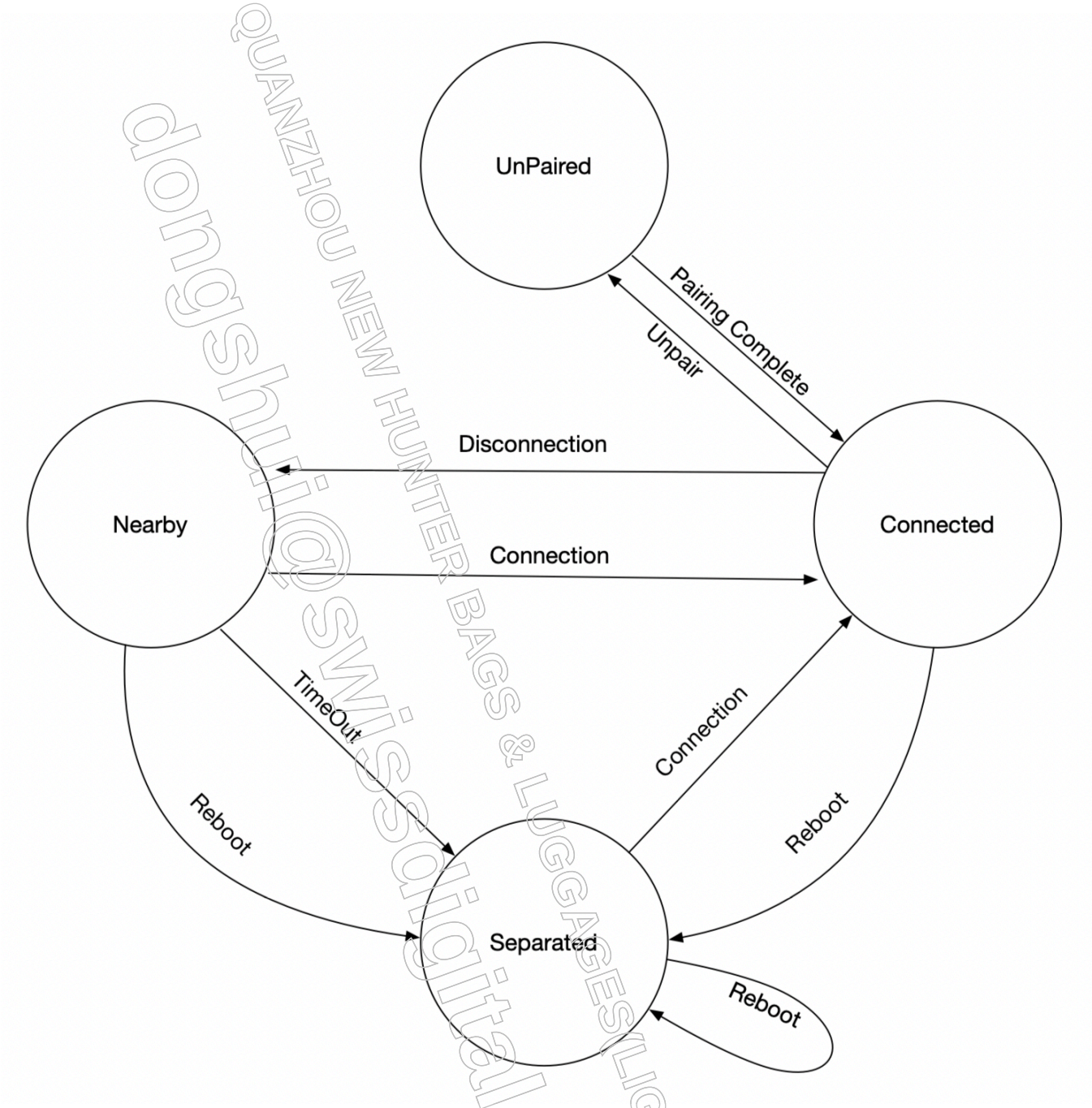

Figure 2-2: Accessory state machine

2.7.1. Unpaired

The accessory must be in an unpaired state on first startup or before the accessory setup is completed. In this state, the accessory must advertise Find My network service as a primary service in a connectable Bluetooth advertisement (See Bluetooth LE advertising). The user initiates pairing from an owner device. See Pairing for the pairing procedure.

2.7.2. Connected

The accessory must enter connected state after the Find My network pairing successfully completes with the owner device. The owner device may disconnect from the accessory after pairing completes. Once paired, the accessory must not pair with another Apple device for Find My network functions. It must stay paired until it successfully completes the unpairing procedure with the owner device.

The accessory must reenter the connected state from nearby or separated state or whenever an owner device connects to the accessory. The accessory shall support simultaneous connections to two Apple devices on the same Apple Account.

Motion detection and Unwanted Tracking protocols are disabled in connected state. When the accessory enters this state, advertising payload is set to the nearby key. A paired accessory must disconnect the Bluetooth LE connection if the link encryption is not completed within 10 seconds.

2.7.3. Nearby

The accessory must enter the nearby state immediately after it disconnects from the last owner device. The accessory shall remain in nearby state for T_{NEARBY}. See Timers and constants.

Motion detection and unwanted tracking detection protocols are disabled in nearby state. When the accessory enters this state, advertising payload is set to the nearby key. See Payload for nearby state for details.

2.7.4. Separated

The accessory must enter the separated state under these conditions:

1. The accessory is paired and starts up from a reset, power cycle, or other reinitialization procedure.
2. The accessory is in nearby state and the T_{NEARBY} time-out has expired.

Motion detection and unwanted tracking detection protocols are enabled in separated state. When the accessory enters this state, advertising payload is set to the primary key or the secondary key. See Payload for separated state for details.

3. Requirements

3.1. Overview

Accessories that support the Find My network accessory protocol must conform to the requirements listed in this chapter, along with any feature-specific requirements contained in other chapters.

3.2. General

An accessory that supports the Find My network accessory protocol must meet these requirements:

- It must incorporate software authentication. See Software authentication for details.
- It must enable the user to set up the accessory using the Apple Find My app, both out of the box and after every factory reset, without requiring additional setup procedures.
- It must be certified and listed as a qualified end product by the Bluetooth SIG.
- It must not be intended, marketed or used to track people or pets.
- It must not be intended, marketed or used to deter theft or recover stolen items.
- It must not be intended, marketed or used for asset tracking or fleet management, including ridesharing.
- It must not operate simultaneously on the Find My network and other finder network, or otherwise implement functionality which may interfere with the security and privacy requirements referenced in this document.
- It must be not be intended, marketed or used for enterprise, business-to-business, government, education, or other commercial or institutional use.
- It must comply with the latest versions of the applicable Licensed Specifications when the accessory supports additional MFi licensed technologies.

3.3. Hardware

3.3.1. Bluetooth

The accessory must use a Bluetooth controller that meets the following requirements:

- LE 2M uncoded PHY
- Data packet length extension

For details, refer to the latest version of *Bluetooth Core Specification Feature Overview.*

The Bluetooth LE transmit power level of the accessory shall be fixed at ≥ +4dBm. The transmit power level is the conducted transmit power.

An accessory with a higher transmit power is visible to finder devices over a larger area. The accessory is more likely to be found, and with more frequent location updates. However, the location will be less certain, so that the owner of a misplaced device will have a larger search area. A lower transmit power will have the opposite trade-off. A product should choose a Bluetooth LE transmit power that matches its objective and performance targets.

3.3.1.1.Accessories that advertise non-Find My network Bluetooth payload

Find My network-enabled accessories that advertise a Bluetooth payload in addition to the Bluetooth payload advertised for the Find My network shall comply with the requirements below.

- Bluetooth Classic accessories shall support GATT over Bluetooth Classic and Bluetooth LE.
- Bluetooth LE accessories shall support
  - LE advertising extensions to advertise other services and payload using advertising sets.
  - Random resolvable addresses and periodic rotation of the addresses and Bluetooth payload used in advertising sets, rotating every 15 minutes. This will deter tracking of the accessory by a non-owner when it is in physical proximity to the owner.
- Accessories that support standard Bluetooth Classic and Bluetooth LE pairing to multiple hosts shall allow Find My network pairing to only one owner.
- When connected to hosts that are not paired to the Find My network, accessories shall expose only paired owner information characteristic of the Find My network service.

Examples of such accessories include Bluetooth headphones and speakers, and accessories that connect to an app.

3.3.1.1.1. Find My device naming

When a Find My network-enabled accessory that advertises a non-Find My network Bluetooth payload is put in Bluetooth pairing mode by the end user, an updated device name that includes the suffix “-Find My” shall be presented in Bluetooth settings. This will indicate that location finding is enabled. The device name is limited to 14 characters total.

This updated device name shall be used in all the name discovery procedures used during pairing and discovery. This includes: Extended Inquiry Response and Remote Name Request procedures for BT Classic devices, Local Name AD type in Advertising, Scan_Response payloads, and device name GATT characteristic for BT LE devices.

When the owner device reconnects to the accessory, the accessory shall revert back to the original device name.

3.3.2. Product-specific requirements

During separated state, motion-triggered UT sound alerts from the accessory are designed to bring awareness to the person who is in proximity to it. These alerts are created using a sound maker (such as a speaker) and a motion detector (such as an accelerometer).

Accessories that would not be easily discovered by the person in proximity to it, must include a sound maker and motion detector to support motion-triggered UT sound alerts. See Unwanted tracking detection for detailed requirements.

Certain accessories that meet the following criteria do not need to include a sound maker or motion detector:

- The Bluetooth module is integrated into the accessory in such a way that it cannot be removed without rendering the Bluetooth module inoperable, AND
- The accessory is larger than 30 cm in at least one dimension, OR
- The accessory is larger than 18 cm X 13 cm in two of its dimensions, OR
- The accessory is larger than 250 cm^3 in three dimensional space

3.3.3. Find My on-product mark

The Locate with Apple Find My icon or the Locate with Apple Find My badge is required on all Find My network-enabled Licensed Products unless any one of these three criteria are met:

- The Licensed Product always requires Bluetooth pairing before its intended use, and has implemented Find My device naming that indicates it is currently trackable by its owner. For example, Bluetooth wireless headphones meet this criteria.

- The Licensed Product is a dedicated tracking device which has no purpose other than to be located using Apple Find My, and it has a motion detector and sound maker. For example, a locator fob that includes motion detection and sound, and is designed to attach to other objects would meet this criteria.

- The Licensed Product is a personal computing device with internet connectivity that is apparent and obvious to the user. For example, computers, tablets, and digital watches that offer internet connectivity meet this criteria.

Refer to the *Works with Apple Find My Identity Guidelines* for additional requirements for the Locate with Apple Find My icon and the Locate with Apple Find My badge.

3.3.4. Serial number requirement

The serial number must be etched, engraved or otherwise directly printed on the accessory, and the end user must be able to easily find it on the accessory. The number must be unique for each product ID. Accessory serial number can be up to 16 bytes of uppercase alphanumeric characters (A-Z, 0-9). If the serial number is less than 16 bytes, trailing null padding shall be added.

3.3.5. Serial number lookup

The serial number must be readable either through NFC tap (see additional requirements under NFC) or Bluetooth LE (see additional requirements under Serial number lookup over Bluetooth LE). For privacy reasons, a serial number must be readable over Bluetooth LE only when a device is paired and upon user action on the accessory (for example, when pressing and holding a button). Buttons should be externally accessible (i.e., not on the circuit board under the battery door). Special tools, like a screwdriver, paperclip, or SIM-removal tool, must not be required for serial number lookup.

Instructions for serial number lookup from the accessory must be provided in the MFi Portal on the Product Plan Find My Data Form.

3.3.6. Find My network disable

The accessory must have a physical mechanism to disable Find My network (for example, a power off button, or battery removal) based on user intent. Special tools, like a screwdriver, paperclip, or SIM-removal tool, shall not be required to disable Find My network.

Instructions for how to disable the Find My network from the accessory must be provided in the MFi Portal on the Product Plan Find My Data Form.

3.3.7. Find My network pairing mode

The accessory must have a physical mechanism to put it into Find My network pairing mode (for example, press a button 3 times) based on user intent. See Pairing mode for additional details.

3.3.8. Reset

The accessory must have a physical mechanism to reset to default factory settings (for example, 5 power cycle attempts within 1 minute) based on user intent. A factory reset must delete all Find My network data except the following:

- Accessory non-owner service
- Firmware version
- Serial number
- Software authentication token
- Software authentication UUID
- Apple server public keys
  - Signature verification key (Q_A)
  - Encryption key (Q_E)

The accessory must reenter Find my network pairing mode when the user initiates it. See Pairing for details.

3.3.9. Clock accuracy

The accessory must support 15-minute key rotation using hardware timers. Apple devices expect the accessory to have a 200 PPM oscillator, causing a potential drift rate of 17.28 seconds per day. See Timers and constants for more details. On every connection, the owner device sends the Configure separated state command. The accessory shall synchronize its clock using the NextPrimaryKeyRoll parameter of the Configure_Separated_Mode command.

3.4. Cryptography

3.4.1. Operations

Pairing the accessory with an owner device as well as deriving keys requires the following:

- A cryptographically secure DRBG (see NIST Special Publication 800-90A) with a reliable source of entropy (see NIST Special Publication 800-90B).
- Modular reduction and addition of big integers.
- An implementation of the SHA-256 cryptographic hash function.
- An implementation of the ANSI x9.63 KDF (see SEC1, 3.6.1 ANSI X9.63 Key Derivation Function).
- Computations on the NIST P-224 elliptic curve (see NIST SP 800-186, 3.2.1.2. P-224):
  - Generation of a random scalar in [1, q).
  - Scalar multiplication and point addition.
  - Verification that a point is on the P-224 elliptic curve.
- ECDSA/ECDH over the NIST P-256 elliptic curve (see NIST SP 800-186, 3.2.1.3. P-256 and Pairing for more details).
- AES-128-GCM encryption and decryption (see NIST Special Publication 800-38D).

3.4.2. Implementation

Cryptographic operations and algorithms must compute on secret values in constant time to defend against timing attacks. Similarly, a secret value (or parts of one) must not be used as a memory offset or as the condition for a branch instruction.

Scalar generation should either use rejection sampling or generate at least 64 more bits than needed so that the bias due to the modular reduction is negligible (see FIPS 186-5, A.2.1 ECDSA Key Pair Generation using Extra Random Bits and A.2.2 ECDSA Key Pair Generation by Rejection Sampling). The scalar must not be generated by simply reducing the minimally required number of random bytes modulo q (the order of the base point) because this leads to a biased distribution.

Implementation of the scalar multiplication and point addition on elliptic curves must be safe against timing attacks. An exception may be made when computing on public values; for example, to speed up ECDSA signature verification. A variable-time, double-base scalar multiplication for ECDSA signature verification must not be used to compute primary or secondary keys.

Upon receiving a scalar, it must be checked to be in range [1, q), where q is the order of the base point of the elliptic curve, before continuing with the protocol flow. See Scalar validation.

Upon receiving an elliptic curve point, it must be checked to be on the curve. See Elliptic curve point validation.

3.4.2.1. Endianness and wire format

All elliptic curve points, coordinates, and scalars must be transmitted in big-endian byte order; that is, the most significant bytes are sent first.

Whenever a scalar or a coordinate is the input for an algorithm like SHA-256() or ANSI-X9.63-KDF(), or the output of a function, its byte order is assumed to be big-endian. A point is expected to be formatted in uncompressed ANSI X9.63 format. See SEC1, 2.3.3 EllipticCurvePoint-to-OctetString Conversion.

3.4.2.2. Random scalar generation

Whenever this specification requires generation of a P-224 scalar, follow this process:

1. Generate r = 28 random bytes using a cryptographically secure DRBG. See Operations.
2. If r >= q - 1, where q is the order of the base point of the P-224 elliptic curve, go to step 1.
3. Compute s = r + 1 and return s as the new scalar.

Another option to securely generate a P-224 scalar is as follows:

1. Generate r = 36 random bytes using a cryptographically secure DRBG. See Operations.
2. Compute k = r (mod q-1), where q is the order of the base point of the P-224 elliptic curve.
3. Compute s = k + 1 and return s as the new scalar.

Whenever this specification requires generation of a P-256 scalar, follow this process:

1. Generate r = 32 random bytes using a cryptographically secure DRBG. See Operations.
2. If r >= q - 1, where q is the order of the base point of the P-256 elliptic curve, go to step 1.
3. Compute s = r + 1 and return s as the new scalar.

Another option to securely generate a P-256 scalar is as follows:

1. Generate r = 40 random bytes using a cryptographically secure DRBG. See Operations.
2. Compute k = r (mod q-1), where q is the order of the base point of the P-256 elliptic curve.
3. Compute s = k + 1 and return s as the new scalar.

3.4.2.3. Scalar validation

Whenever this specification requires validation of a P-224 scalar, follow this process:

1. If the given scalar s = 0, reject it as invalid.
2. If s >= q, where q is the order of the base point of the P-224 elliptic curve, reject s as invalid.
3. Make s a valid scalar.

3.4.2.4. Elliptic curve point validation

Whenever this specification requires validation of a P-224 elliptic curve point, follow this process:

1. Check that the length of a point is 57 bytes.
2. Decode x and y as big-endian integers in the range [0, 2^{224}).
3. Check that x < p and y < p, where p = 2^{224} - 2^{96} + 1.

4. Check that $y^2 = x^3 + ax + b$, where a = p - 3 and b = 0xb4050a850c04b3abf54132565044b0b7d7bfd8ba270b39432355ffb4.

3.4.2.5. ECDSA signature verification

Whenever this specification requires verification of a P-256 ECDSA signature over a message m:

1. Decode the given signature to obtain two 32-byte big-endian integers r and s (see [SEC1, C.5 Syntax for Signatures](#)).
2. Check that 0 < r < p and 0 < s < p, where $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.
3. Compute e = SHA-256(m), where m is the signed message.
4. Let z be the |q| leftmost bits of e, where |q| is the bit length of the group order q.
5. Compute $u_1 = zs^{-1}$ (mod q) and $u_2 = rs^{-1}$ (mod q).
6. Compute the point $(x, y) = u_1 \cdot G + u_2 \cdot Q_A$, where G is the base point of the P-256 elliptic curve and Q_A is Apple's signature verification key.
7. If (x, y) is the point at infinity, reject the signature.
8. If r = x (mod q), then accept the signature, and if not, reject it.

See Apple server public keys for signature verification key (Q_A) details.

3.4.2.6. ECIES encryption

Whenever this specification requires encryption of a message M to a P-256 public key $P=Q_E$ (Apple server encryption key), follow this process:

1. Generate an ephemeral P-256 scalar k as described in Random scalar generation.
2. Compute the public point $Q = k \cdot G$, where G is the base point of the P-256 elliptic curve.
3. Compute the shared secret $Z = k \cdot P$.
4. Derive 32 bytes of keying material as V = ANSI-X9.63-KDF(x(Z), Q || P).
5. Set K = V[0..15], that is, the first 16 bytes of the keying material V.
6. Set IV = V[16..31], that is, the last 16 bytes of the keying material V.
7. Encrypt message M as (C,T) = AES-128-GCM(K, IV, M) without any additional authenticated data. K is the 128-bit AES key, IV is the initialization vector, C is the ciphertext, and T is the 16-byte authentication tag.
8. Output Q || C || T; that is, the ephemeral public key Q concatenated with the ciphertext and the authentication tag.

See Apple server public keys for the Apple server's encryption key (Q_E) details.

3.4.2.7. AES-GCM decryption

Whenever this specification requires AES-128-GCM decryption of a message M, given a 128-bit AES key K and a 16-byte IV, follow this process:

1. Decode message C in the following way: With n = length(C), take the first n - 16 bytes as the cipher text C. The last 16 bytes are the authentication tag T.

2. Decrypt ciphertext C as (M,T') = AES-128-GCM(K, IV, C) without any additional authenticated data.
3. Compare authentication tags T and T'. Do not abort as soon as a mismatch is found, but report an error only after all bytes have been compared.
4. If T ≠ T', abort and discard the ciphertext.

3.4.2.8. Random generation

Whenever this specification requires generation random values, a cryptographically secure DRBG must be used.

3.5. Software authentication

Refer to the *Software Authentication Server Specification* on how to obtain software tokens.

- A software authentication token, along with its corresponding UUID, must be provisioned on the accessory through factory provisioning (at the time of accessory manufacturing and firmware flashing).
- The software authentication token and its UUID must be decoded using Base64 from the file provided by Apple's server and stored as raw data bytes on the accessory.
- The UUID associated with the software authentication token must be registered with Apple server as defined in the *Software Authentication Server Specification* after provisioning on the accessory.
- The software authentication token must be stored in secure storage on the accessory.
- The provisioned software authentication token must persist through factory reset.
- The provisioned software authentication token is for one-time use only. The software authentication token and corresponding UUID will be required during pairing as part of the Find My network pairing process. A new software authentication token will be provided to the accessory during pairing and must be stored by the accessory for future use. See Pairing and key management for more details.

3.6. Apple server public keys

Whenever this specification requires Apple server signature verification, the accessory must use the Apple server signature verification key (Q_A).

Whenever this specification requires encryption to Apple server, the accessory must use the Apple server encryption key (Q_E).

- Apple server public keys must be provisioned in the accessory through factory provisioning (at the time of accessory manufacturing and firmware flashing) with integrity protection.
- Apple server public keys must be stored in secure storage on the accessory and protected against tampering.

MFi Licensees will have access to Apple server public keys.

3.7. Power cycle

A user may power cycle an accessory for various reasons (for example, battery replacement or user restart). When an accessory is power cycled, it shall start in separated state with the current secondary key as the separated key. See After power cycle for advertisement details.

3.8. Firmware updates

Accessories must support firmware updates. See Firmware update for more details.

- All firmware images must be authenticated and verified by the accessory using a mechanism that guarantees the integrity of the image from the manufacturer.
- Updated firmware must complete MFi certification requirements before release.
- Accessories must not allow a firmware image to be downgraded after a successful firmware update.

4. Bluetooth Requirements

4.1. Overview

Bluetooth Low Energy (LE) is used as the wireless transport for all communication between Apple products and accessories.

4.2. Bluetooth advertising

The accessory should advertise the Find My network payload at the $T_{FMN\_ADV\_INTERVAL}$ interval in a connectable advertising ADV_IND PDU to match the Bluetooth LE scan duty cycles of the Apple device. The accessory may advertise other Advertising Data Type (AD Type) in other advertising events. The accessory shall continue advertising the Find My network payload until all owner devices are connected, up the maximum number of connections. See Set max connections for details.

4.3. Bluetooth connection

The accessory must support at least two simultaneous connections in a peripheral role.

The connection interval of the Bluetooth LE link between the Apple device and accessory depends on the type of user interaction. An Apple device typically selects a connection interval in multiples of 15 ms. The accessory shall support a connection interval that is a multiple of 15 ms. The Apple device may use 990 ms as the Bluetooth LE connection interval to the accessory.

4.4. Bluetooth host

4.4.1. Services

The Find My network service and Accessory non-owner service must be instantiated as primary services. The accessory must also support the following services:

- Tx Power service

The accessory must set the Tx power Level characteristic to the Bluetooth LE TRP (Total radiated power). See discussion on Bluetooth LE transmit power for details.

4.4.2. MTU size

The accessory shall select a MTU size that is equal to or greater than the MTU request from the Apple device.

4.4.3. Link encryption key

The accessory pairs to the Apple device using the Bluetooth LE Just Works pairing scheme. Once Bluetooth LE paired, the Apple device initiates the Find My network pairing procedure. See Pairing for more details. To encrypt the Bluetooth LE link on every subsequent connection, the accessory must use the LTK generated by the Find My network protocol. See LTK generation for details on LTK generation and use.

4.4.4. Handling concurrent operations

An app on the Apple device may interact with the accessory over GATT or, if supported, connection-oriented L2CAP channels. Apple devices may connect and perform Find My network GATT operations independently from other interactions with the accessory.

The accessory shall support Find My network GATT interactions while simultaneously supporting GATT and connection-oriented L2CAP channels from other Apple devices.

4.4.5. Time-out

Unless otherwise specified, the accessory must respond to all control point commands within 30 seconds.

4.4.6. Indications

Unless otherwise specified, the accessory shall send indications only to requesting connection.

4.5. Accessory non-owner service

4.5.1. Service

The Accessory non-owner service UUID is 15190001-12F4-C226-88ED-2AC5579F2A85. This service shall use GATT over LE and, if available, Bluetooth Classic transport. The accessory non-owner service shall be instantiated as a primary service.

The values of the following accessory non-owner service - characteristics must be persistent through the lifetime of the accessory, Product data, Manufacturer name, Model name, Accessory category, Accessory capability, and Battery type. These values must match with the data as specified in the MFi Portal at the time of firmware submission for self-certification. Note: Some of these values may be visible to the user in the accessory settings of the Apple Find My app.

4.5.2. Byte transmission order

All characteristics used with this service shall be transmitted with the least significant octet first (that is, little endian).

4.5.3. Characteristics

The UUID for Accessory non-owner service characteristics is 8E0C0001-1D68-FB92-BF61-48377421680E.

Table 4-1 Accessory non-owner service - characteristics

Opcode	Opcode Value	Operands	GATT subprocedure	Direction	Requirement
ProductData	0x0003	None	Write	To accessory	Mandatory
ProductDataResponse	0x0803	Product data	Indications	From accessory	Mandatory
ManufacturerName	0x0004	None	Write	To accessory	Mandatory
ManufacturerNameResponse	0x0804	Manufacturer name	Indications	From accessory	Mandatory
ModelName	0x0005	None	Write	To accessory	Mandatory
ModelNameResponse	0x0805	Model Name	Indications	From accessory	Mandatory
AccessoryCategory	0x0006	None	Write	To accessory	Mandatory
AccessoryCategoryResponse	0x0806	Accessory Category	Indications	From accessory	Mandatory
ProtocolImplementationVersion	0x0007	None	Write	To accessory	Mandatory
ProtocolImplementationVersionResponse	0x0807	Protocol Implementation Version	Indications	From accessory	Mandatory
AccessoryCapabilities	0x0008	None	Write	To accessory	Mandatory
AccessoryCapabilities-Response	0x0808	Accessory Capabilities	Indications	From accessory	Mandatory
NetworkID	0x0009	None	Write	To accessory	Mandatory
NetworkIDResponse	0x0809	Network ID	Indications	From accessory	Mandatory
FirmwareVersion	0x000A	None	Write	To accessory	Mandatory
FirmwareVersionResponse	0x080A	Firmware Version	Indications	From accessory	Mandatory

Opcode	Opcode Value	Operands	GATT subprocedure	Direction	Requirement
BatteryType	0x000B	None	Write	To accessory	Mandatory
BatteryTypeResponse	0x080B	Battery Type	Indications	From accessory	Mandatory
BatteryLevel	0x000C	None	Write	To accessory	Optional
BatteryLevelResponse	0x080C	Battery Level	Indications	From accessory	Optional
FindMyVersion	0x000D	None	Write	To accessory	Mandatory
FindMyVersionResponse	0x080D	Find My Version	Indications	From accessory	Mandatory

4.5.3.1. Product data

The ProductData operand represents the 8 byte product data value assigned to each Product Plan in the MFi Portal upon Product Plan submission.

Product data received from MFi Portal is a 16 character string. It is composed of two 8 character hex strings (lowercase zero padded), each of which will give you 4 bytes. That makes the total 8 bytes to be sent as Product data.

Table 4-2 Product Data

Operand name	Data type	Size (octets)	Description
Product Data	Uint8	8	ProductData

For e.g. the Product data value of dfeceff1e1ff54db, the value converted to binary would be

dfeceff1 11011111 11101100 11101111 11110001

e1ff54db 11100001 11111111 01010100 11011011

4.5.3.2. Manufacturer name

The manufacturer name operand contains the name of the company whose brand will appear on the accessory, e.g., ”Acme”.

Table 4-3 Manufacturer name

Operand name	Data type	Size (octets)	Description
Manufacturer name	UTF-8	64	Manufacturer name

When the Manufacturer Name is less than 64 bytes, it SHALL be formatted either as:

- A string value with length less than 64 bytes.
- A string value that is both zero-terminated and zero-padded up to 64 bytes.

4.5.3.3. Model name

The model name operand contains the manufacturer specific model of the accessory.

Table 4-4 Model Name

Operand name	Data type	Size (octets)	Description
Model Name	UTF-8	64	Model name

When the Model Name is less than 64 bytes, it SHALL be formatted either as:

- A string value with length less than 64 bytes.
- A string value that is both zero-terminated and zero-padded up to 64 bytes.

4.5.3.4.Accessory category

The accessory category operand describes the category the accessory most closely resembles.

Table 4-5 Accessory category

Operand name	Data type	Size (octets)	Description
Accessory category	Uint8	8	See Accessory Category for details Byte 0: Uint8 value of Accessory Category Byte 1-7: Reserved

4.5.3.5.Protocol Implementation Version

The Protocol Implementation version operand describes the current implementation version of the protocol.

Table 4-6 Protocol Implementation Version

Operand name	Data type	Size (octets)	Description
ProtocolImplementationVersion	Uint32	4	Byte 0 : revision version number Byte 1 : minor version number Byte 2:3 : major version number This must be set to 1.0.0 for this version of the specification. The equivalent 4-byte value is 0x00010000

4.5.3.6.Accessory capabilities

The accessory capabilities operand describes the various Find My network protocol capabilities supported on the accessory.

Table 4-7 Accessory capability

Operand name	Data type	Size (octets)	Description
Accessory capabilities	Uint32	4	Bit 0 : Supports play sound Bit 1 : Supports motion detector UT Bit 2 : Supports serial number lookup by NFC Bit 3 : Supports serial number lookup by BLE Bit 4: Supports firmware update service Bits 5-31: Reserved

For e.g. an accessory supporting playSound, NFC, UT and firmware update service will have the value set as:

00000000 00000000 00000000 00010111 in binary

0x00000017 as Hex

23 as UInt32

4.5.3.7.Firmware version

The Firmware version operand describes the current firmware version on the product.

The firmware revision string shall use the x[.y[.z]] format where :

• <x> is the major version number, required.

• <y> is the minor version number, required if it is non zero or if <z> is present.

• <z> is the revision version number, required if non zero.

The firmware revision must follow these rules:

• <x> is incremented when there is significant change; for example, 1.0.0, 2.0.0, 3.0.0, and so on.

• <y> is incremented when minor changes are introduced, such as 1.1.0, 2.1.0, 3.1.0, and so on.

• <z> is incremented when bug fixes are introduced, such as 1.0.1, 2.0.1, 3.0.1, and so on.

• Subsequent firmware updates can have a lower <y> version only if <x> is incremented.

• Subsequent firmware updates can have a lower <z> version only if <x> or <y> is incremented.

• Major version must not be greater than (2^16 -1).

• Minor and revision version must not be greater than (2^8 -1).

• The characteristic value must change after every firmware update.

Table 4-8 Firmware version

Operand name	Data type	Size (octets)	Description
FirmwareVersion	Uint32	4	Byte 0 : revision version number Byte 1 : minor version number Byte 2:3 : major version number

As an example, a Major.Minor.Revision value of 1.0.0 has an equivalent 4-byte value of 0x00010000.

4.5.3.8.Find My network version

The Find My network version operand indicates the Find My network specification version the product complies with. The version format matches the firmware version format described in the previous section.

Table 4-9 Find My network version

Operand name	Data type	Size (octets)	Description
Find My network version	Uint32	4	Byte 0 : revision version number Byte 1 : minor version number Byte 2:3 : major version number This must be set to 2.0.0 for this version of the specification. The equivalent 4-byte value is 0x00020000

4.5.3.9.Battery type

The Battery type operand describes the battery type used in the accessory.

Table 4-10 Battery type

Operand name	Data type	Size (octets)	Description
Battery Type	Uint8	1	0 = Powered 1 = Non-rechargeable battery 2 = Rechargeable battery

4.5.3.10.Battery level

The Battery level operand indicates the current battery level.

Table 4-11 Battery state

Operand name	Data type	Size (octets)	Description
Battery Level	Uint8	1	Battery state definition 0 = Full 1 = Medium 2 = Low 3 = Critically low

4.5.3.11.Network ID

The 1-byte Network ID operand indicates a registered value for the manufacturer, as defined in Manufacturer Registry.

Table 4-12 Manufacturer registry

Operand Name	Data type	Size (octets)	Description
Network ID	Uint8	1	This must be set to 0x01 for accessories operating on the Find My network.

4.5.3.12. Non-owner control point

The non-owner control point enables a non-owner device to locate the accessory by playing a sound. The non-owner control point shall use the same service and characteristic as defined in 4.5.1 Accessory non-owner service. The opcodes for the control point are defined in Table 4-13.

Table 4-13 Non-owner control point

Opcode	Opcode value	Operands	GATT subprocedure	Direction
Sound_Start	0x0300	None	Write	To accessory
Sound_Stop	0x0301	None	Write	To accessory
Command Response	0x0302	CommandOpCode ResponseStatus	Indications	From accessory
Sound_Completed	0x0303	None	Indications	From accessory
Get_Serial_Number	0x0404	None	Write	To accessory
Get_Serial_Number_Response	0x0405	SerialNumberPayload	Indications	From accessory

This control point shall be available to the non-owner device only when the accessory is in separated state. In all other states, the accessory shall return the Invalid_command error as the responseStatus in CommandResponse. See Command Response for details.

4.5.3.13.Non-owner control point procedures

The accessory, as server, shall indicate the non-owner control point for responding to the commands from the non-owner device.

4.5.3.13.1. Play sound—Non-owner control point

Play sound requirements are applicable only to accessories that include a sound maker. See Product-specific requirements.

The Sound_Start opcode is used to play sound on the sound maker of the accessory. The sound maker must play sound for a minimum duration of 5 seconds.

The accessory shall confirm the start of the play sound procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

Once the play sound action is completed, the accessory sends the Sound_Completed message.

The Sound_Stop opcode is used to stop an ongoing sound request.

If the sound event is completed or was not initiated by the non-owner device, the accessory responds with the Invalid_state ResponseStatus code.

If the accessory does not support the play sound procedure, it responds with Invalid_command ResponseStatus code.

If a Sound_Start procedure is initiated when another play sound action is in progress, it rejects with Invalid_state error code.

The accessory shall confirm the completion of the stop sound procedure by sending the Sound_Completed message.

4.5.3.13.2. Get Serial Number

The Get_Serial_Number is used to retrieve serial number lookup payload over Bluetooth LE. This must be enabled for $T_{FMN\_SEPARATED\_SN\_LOOKUP\_INTERVAL}$ duration upon user action on the accessory (for example, press and hold a button for 10 seconds to initiate serial number read state). When the accessory is in this mode, it must respond with Get_Serial_Number_Response.

Table 4-14 Get Serial Number Response

Operand	Data type	Size (octets)	Description
SerialNumber-Palyload	bytes	141	Encrypted serial number (e) when in paired state. See Serial number payload information for details.

If the accessory is not in serial number read state, it must send Command Response with a status of Invalid_state.

4.6. Find My network service

4.6.1. Service

The Find My network service UUID is 0xFD44. This service shall use GATT over LE and, if available, Bluetooth Classic transport.

4.6.2. Byte transmission order

All characteristics used with this service shall be transmitted with the least significant octet first (that is, little endian).

4.6.3. Characteristics

The UUID for Find My network service characteristics is 4F86XXXX-943B-49EF-BED4-2F730304427A, where XXXX is unique for each characteristic.

Table 4-15 Find My network service - characteristics

Characteristic name	UUID	Requirement	Mandatory properties	Security permissions
Pairing Control Point	0x0001	Mandatory	Write, Indicate	Authorization not Required

Characteristic name	UUID	Requirement	Mandatory properties	Security permissions
Configuration Control Point	0x0002	Mandatory	Write, Indicate	Authorization required
Reserved	0x0003	N/A	N/A	N/A
Paired owner Informa-tion Control Point	0x0004	Mandatory	Write, Indicate	Authorization not required
Debug Control Point	0x0005	Mandatory	Write, Indicate	Authorization not required

A client characteristic configuration descriptor shall be included for all the characteristics, as required.

4.6.3.1.Pairing control point

The pairing control point enables you to pair an accessory with an owner device. The opcodes for the control point is defined in Table 4-16.

Table 4-16 Pairing control point opcodes

Opcode	Opcode value	Operands	GATT subprocedure	Direction
Initiate_pairing	0x0100	SessionNonce E1	Write	To accessory
Send_pairing_data	0x0101	C1 E2	Indications	From accessory
Finalize_pairing	0x0102	C2 E3 SeedS S2 iCloudIdentifier	Write	To accessory
Send_pairing_status	0x0103	C3 Status E4	Indications	From accessory
Pairing_complete	0x0104	NextPrimaryKeyRoll	Write	To accessory

4.6.3.2. Pairing control point procedures

The accessory, as server, shall indicate the pairing control point for responding to the commands from the Apple device.

4.6.3.2.1.Initiate pairing

The Initiate_pairing opcode is used to start the pairing session of an accessory from an Apple device.

Table 4-17 Initiate pairing

Operand	Data type	Size (octets)	Description
SessionNonce	bytes	32	Nonce generated by owner device
E1	bytes	113	Encrypted blob generated by owner device

4.6.3.2.2.Send pairing data

The Send_pairing_data opcode must be used by the accessory to respond to a pairing session request. The accessory must respond in 60 seconds.

Table 4-18 Send pairing data

Operand	Data type	Size (octets)	Description
C1	bytes	32	Data sent by the accessory as initial commitment for pairing (see Collaborative key generation for C1 details)
E2	bytes	1326	Encrypted blob generated by accessory

See Send pairing data for E2 generation details.

4.6.3.2.3.Finalize pairing

The Finalize_pairing opcode is used by an Apple device to confirm pairing. See Validate and confirm pairing for more details.

Table 4-19 Finalize pairing

Operand	Data type	Size (octets)	Description
C2	bytes	114	Shared commitment generated by Apple device
E3	bytes	1040	Encrypted software token that's vended by Apple server for each accessory

Operand	Data type	Size (octets)	Description
SeedS	bytes	32	Unique server seed for each accessory that's paired
iCloud_Identifier	bytes	60	masked Apple Account
S2	bytes	100	Apple server signature to confirm pairing

4.6.3.2.4.Send pairing status

The Send_pairing_status opcode must be used by the accessory to confirm pairing.

Table 4-20 Send pairing status

Operand	Data type	Size (octets)	Description
C3	bytes	85	Final commitment generated by accessory. See Collaborative key generation for C3 details.
Status	UInt32	4	0 - for success 1 - error signature verification 2 - error saving data
E4	bytes	1286	Encrypted blob generated by accessory

See Send pairing status for E4 generation details.

4.6.3.2.5.Pairing complete

The Pairing_complete opcode is used to complete pairing the accessory from the Apple device. The valid range for nextPrimaryKeyRoll parameter is 0 to 15 minutes.

Table 4-21 Pairing complete

Operand	Data type	Size (octets)	Description
NextPrimaryKey-Roll	UInt32	4	Time in milliseconds until the next primary key rotation

4.6.3.3. Configuration control point

The configuration control point enables you to configure Find My network functionality on the accessory and enable Find My network interactions. The opcodes for the control point are defined in Table 4-22.

Table 4-22 Configuration control point opcodes

Opcode	Opcode Value	Operands	GATT subprocedure	Direction
Sound_Start	0x0200	None	Write	To accessory
Sound_Stop	0x0201	None	Write	To accessory
Persistent_ Connection_Status	0x0202	Persistent Connection-Status	Write	To accessory
Set_Nearby_Timeout	0x0203	NearbyTimeOut	Write	To accessory
Unpair	0x0204	None	Write	To accessory
Configure_Separated_S-tate	0x0205	NextKeyRoll SecondaryKeyEvaluationIndex	Write	To accessory
Latch_Separated_Key	0x0206	None	Write	To accessory
Set_Max_Connections	0x0207	MaxConnections	Write	To accessory
Set_UTC	0x0208	CurrentTime	Write	To accessory
Get_Multi_Status	0x0209	None	Write	To accessory
Keyroll_Indication	0x020A	KeyIndex	Indications	From accessory
Command_Response	0x020B	CommandOpCode ResponseStatus	Indications	From accessory
Get_Multi_Status_Response	0x020C	MultiStatus	Indications	From accessory
Sound_Completed	0x020D	None	Indications	From accessory
Latch_Separated_Key_Response	0x020E	LatchedPrimaryKeyIndex	Indications	From accessory
Get_Firmware_Version	0x0229	None	Write	To accessory
Get_Firmware_Version_Response	0x022A	Firmware Version	Indications	From accessory

4.6.3.4.Configuration control point procedures

The paired Apple device initiates a configuration procedure using one of the messages defined in Table 4-22. The accessory, as server, shall indicate the Configuration control point for responding to the commands from the Apple device.

4.6.3.4.1.Play sound—owner control point

Play sound requirements are applicable only for accessories that include a sound maker. See Product-specific requirements.

The Sound_Start opcode is used to play sound on the sound maker of the accessory.

The accessory shall confirm the start of the play sound procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

Once the play sound action is completed, the accessory sends the Sound_Completed message.

The Sound_Stop opcode is used to stop an ongoing sound request.

If the sound event is completed or was not initiated by the Apple device, the accessory responds with Invalid_state ResponseStatus code.

If the accessory does not support the play sound procedure, it responds with Invalid_command ResponseStatus code.

If a Sound_Start procedure is initiated when another play sound action is in progress, it rejects with Invalid_state error code.

The accessory shall confirm the completion of the stop sound procedure by sending the Sound_Completed message.

4.6.3.4.2.Persistent connection status

The Persistent_Connection_Status opcode is used by the owner device to indicate whether the accessory is persistently connected using an always-connected Bluetooth LE link.

If persistent connection is enabled, on a link lost event, the accessory shall advertise using an advertising interval of $T_{RECONNECT\_ADV\_INTERVAL}$ for a duration of $T_{RECONNECT\_ATTEMPT\_TIMEOUT}$.

The accessory shall confirm the completion of the procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

Table 4-23 Persistent connection status

Operand	Data type	Size (octets)	Description
Persistent_ Connection_Status	Boolean	1	0: Persistent connection disabled 1: Persistent connection enabled

4.6.3.4.3.Set nearby timeout

The Set_Nearby_Timeout opcode is used by the owner device to set the time duration to transition from nearby state to separated state, T_{NEARBY}. The valid range for the NearbyTimeOut parameter is 0 to 3600 seconds. A NearbyTimeOut of 0 seconds indicates an immediate transition to separated state.

Table 4-24 Set NearbyTimeOut

Operand	Data type	Size (octets)	Description
NearbyTimeOut	Uint16	2	TimeOut in seconds

The accessory shall confirm the completion of the procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

4.6.3.4.4. Unpair

The Unpair opcode is used to unpair the accessory from the Apple device. This opcode has no parameters. If the accessory is connected to more than one Central device, the accessory shall reject the unpair procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Invalid state. See Unpair procedure for details. The accessory shall confirm the completion of the procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

4.6.3.4.5. Configure separated state

The Configure_Separated_State opcode is sent on every connection. The valid range for nextPrimaryKeyRoll parameter is 0 to 15 minutes.

The valid range for secondaryKeyEvaluationIndex parameter is CurrentPrimaryKeyIndex - 4 to currentPrimaryKeyIndex + 96. The accessory shall confirm the completion of the procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

Table 4-25 Configure separated state

Operand	Data type	Size (octets)	Description
NextPrimaryKeyRoll	Uint32	4	Time in milliseconds until the next primary key rotation
SecondaryKeyEvaluation-Index	Uint32	4	Primary key index at which secondary key is re-evaluated. This corresponds to 4 a.m. local time.

4.6.3.4.6.Latch separated key

The Latch_Separated_Key opCode instructs the accessory to use the current primary key as Separated key until the next 4 a.m. local time. This message has no parameters. The accessory shall confirm the completion of the procedure by sending a Latch_Separated_Key_Response with the latched Primary Key index.

Table 4-26 Latch Separated Key Response

Operand	Data type	Size (octets)	Description
LatchedPrimaryKeyIndex	Uint32	4	Latched primary key index

4.6.3.4.7.Set max connections

The Set_Max_Connections opcode is used to set the maximum number of Bluetooth connections that must be supported by the accessory. Accessories shall support at least two simultaneous Bluetooth connections. If MaxConnections parameter is greater than the accessory's connection limit, the accessory shall set the maxConnections to its maximum supported connections limit. If MaxConnections is set to 1, the accessory shall disconnect all the other connections. The accessory shall confirm the completion of the procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

Table 4-27 Set max connections

Operand	Data type	Size (octets)	Description
MaxConnections	Uint8	1	Maximum Bluetooth connections to be supported by accessory

4.6.3.4.8.Set UTC

The Set_UTC opcode is used to set the current UTC time on the accessory. The accessory shall confirm the completion of the procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

Table 4-28 Set UTC

Operand	Data type	Size (octets)	Description
CurrentTime	Uint64	8	Current UTC time in ms (Jan 1 2001 Epoch)

4.6.3.4.9.Keyroll indication

The Keyroll_Indication opcode must be used by the accessory to indicate that a primary key roll has occurred. The accessory shall send this indication to all the connected Find My network paired devices.

Table 4-29 Keyroll indication

Operand	Data type	Size (octets)	Description
KeyIndex	Uint32	4	Current primary key index

4.6.3.4.10.Command response

The CommandOpCode indicates the procedure that the accessory is responding to, and ResponseStatus indicates the status of the response.

Table 4-30 Command response

Operand	Data type	Size (octets)	Description
CommandOpCode	Uint16	2	The control procedure matching this response
ResponseStatus	Uint16	2	0x0000 Success 0x0001 Invalid_state 0x0002 Invalid_configuration 0x0003 Invalid_length 0x0004 Invalid_param 0xFFFF Invalid_command

4.6.3.4.11.Get multi status response

The Get_Multi_Status_Response opcode must be used by the accessory to respond to the Get_Multi_Status command from the Apple device. The multiStatus is a bit mask that indicates the current state of the accessory. Setting a bit in MultiStatus indicates that the accessory is in that state.

Table 4-31 Multistatus

Operand	Data type	Size (octets)	Description
MultiStatus	Uint32	4	Bit0: Multi_Status_Persistent_Connection Bit1: Reserved Bit2: Multi_Status_Playing_Sound Bit3: Multi_Status_Updating_Firmware Bit4: Reserved Bit5: Reserved Bit6: Multi_Status_Multiple_Owners_Currently_Connected Bit7-31: Reserved

4.6.3.5. Paired owner information control point

The pairing status control point enables an Apple device to read the Find My network pairing status of the accessory.

Table 4-32 Paired owner information control point

Opcode	Opcode value	Operands	GATT subprocedure	Direction
Get_Current_Primary_Key	0x0400	None	Write	To accessory
Get_iCloud_Identifier	0x0401	None	Write	To accessory
Get_Current_Primary_Key_Response	0x0402	Current_Primary_key	Indications	From accessory
Get_iCloud_Identifier_Response	0x0403	iCloud_Identifier	Indications	From accessory
Reserved	0x0404	N/A	N/A	N/A
Reserved	0x0405	N/A	N/A	N/A
Command_Response	0x0406	CommandOpCode ResponseStatus	Indications	From accessory

4.6.3.6.Paired owner information control point procedures

4.6.3.6.1.Get Current Primary Key

The Get_Current_Primary_Key is used to retrieve the currently used Primary Key. If the Find My network pairing is not complete, the accessory shall respond with Get_Current_Primary_Key_Response with Current_Primary_Key set to zero. If the Find My network pairing is complete, the accessory shall respond with Get_Current_Primary_Key_Response with Current Primary Key.

Table 4-33 Current primary key

Operand	Data type	Size (octets)	Description
Current_Primary_key	Uint8	28	Current Primary key, P_i

4.6.3.6.2.Get iCloud Identifier

The Get_iCloud_Identifier is used to retrieve the iCloud identifier associated with Find My network pairing. If the Find My network pairing is not complete, the accessory shall respond with Get_iCloud_Identifier_Response with iCloud_Identifier set to zero (all 60 bytes). If the Find My network pairing is complete, the accessory shall respond with Get_iCloud_Identifier_Response with the iCloud identifier.

Table 4-34 iCloud identifier

Operand	Data type	Size (octets)	Description
iCloud_Identifier	bytes	60	iCloud identifier set during Find My network pairing

4.6.3.6.3. Command Response

The accessory shall respond to any invalid opcode with Command_Response with the Invalid_command error as the responseStatus. See Command Response for details.

4.6.3.7.Debug control point

The debug control point enables you to debug the accessory during development. This control point shall not be enabled in shipping firmware.

The opcodes for the control point are defined in Table 4-35.

Table 4-35 Debug control point

Opcode	Opcode value	Operands	GATT subprocedure	Direction
Set_Key_Rotation_Timeout	0x0500	Timeout	Write	To accessory
Retrieve_Logs	0x0501	None	Write	To accessory
Log_Response	0x0502	LogResponse	Indications	From accessory
Command Response	0x0503	CommandOp-Code ResponseStatus	Indications	From accessory
Reset	0x0504	None	Write	To accessory
UT_Motion_Timers_Config	0x0505	SeparatedUT-TimeoutSeconds SeparatedUT-BackoffTimeout-Seconds	Write	To accessory

4.6.3.8.Debug control point procedures

This control point shall be available only when the accessory is in development. In all other states, the accessory shall return the Invalid_command error as the responseStatus in CommandResponse.

4.6.3.8.1.Set key rotation time-out

The Set_Key_Rotation_Timeout debug command accelerates key rotation by configuring a short time-out. The accessory shall confirm the completion of the procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

Table 4-36 Set key rotation timeout

Operand	Data type	Size (octets)	Description
Timeout	Uint32	4	Time in milliseconds until the next primary key rotation

4.6.3.8.2.Retrieve logs

The Retrieve_Logs debug command is used to dump logs from an accessory. The logs are encoded in UTF-8 format. The accessory transfers the logs with multiple Log_Response Indications. The size of the Log_Response is limited by the MTU size negotiated during connection setup. The accessory indicates the end of the log dump by sending a Log_Response with empty payload.

4.6.3.8.3.Reset

The Reset command is used to reset the accessory. The accessory reboots after confirming the reset procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

4.6.3.8.4.UT motion timers config

The UT_Motion_Timers_Config debug command configures $T_{SEPARATED\_UT\_TIMEOUT}$ and $T_{SEPARATED\_UT\_BACKOFF}$. The accessory shall confirm the completion of the procedure by sending a Command_Response with the corresponding CommandOpCode and a ResponseStatus value of Success.

Table 4-37 UT Motion Timers Config

Operand	Data type	Size (octets)	Description
SeparatedUTTimeout-Seconds	Uint32	4	Time in seconds accessory must be in Separated mode before entering UT motion detection mode.
SeparatedUTBackoff-TimeoutSeconds	Uint32	4	Time in seconds accessory must be in Separated mode after UT motion detection session ends before reentering UT motion detection session.

4.7. Firmware update service

4.7.1.Service

This service is required if the accessory is using unified accessory restore protocol to update its firmware. See Firmware update for details.

The UUID for firmware update service is 0xFD43.

4.7.2.Byte transmission order

All data transmitted and received on the characteristics of this service shall be interpreted with little endian byte ordering.

4.7.3.Characteristics

This service has one characteristic with UUID 94110001-6D9B-4225-A4F1-6A4A7F01B0DE.

Table 4-38 Firmware update service - characteristics

Characteristic name	Requirement	Mandatory properties	Security permissions
Data Control Point	Mandatory	Write, Indicate	Authorization Required

A client characteristic configuration descriptor is required for all characteristics.

4.7.3.1.Data control point

The data control point characteristic allows the accessory to exchange Unified Accessory Restore Protocol data with the owner device. The interpretation of this data is detailed in the *Unified Accessory Restore Protocol Development Guide*.

Table 4-39 Data control point

Characteristic name	Data type	Size (octets)	Description
Data Control Point	Bytes	Variable	See Unified Accessory Restore Protocol Development Guide for format.

4.8. Fragmentation and reassembly

Characteristics in Find My network service and Firmware update service use payloads that are bigger than the maximum MTU size. To support read, write and indicate operations on such characteristics, the host shall support fragmentation and the data packet format outlined in Table 4-40 for Find My network service and Firmware update service.

Table 4-40 Data packet format

Header Byte 0	Payload Byte 1..MTU-1
Bit 0: Fragment type 0 — Continuation or start of a fragment 1 — Final fragment Bit 1:7 Reserved	Data payload

If an operation transfers data that is less than the MTU size, the host sets the fragment type in the header to 1. If an operation transfers data that is greater than or equal to the MTU size, the host breaks up the data into multiple fragments and transmits each fragment. The last fragment has the fragment type in the header set to 1. The receiving host reassembles the data payload based on the headers from the received fragments. The host must ensure all fragments of a message are transferred before transmitting the next message.

4.9. Service availability

This table outlines the services and characteristics availability based on the state of the accessory. If a characteristic is unavailable for a given accessory state, the accessory shall respond to any read or write request with Invalid_state error as the responseStatus in CommandResponse. See Command Response for details.

Table 4-41 Service availability

Service	Characteristic	Accessory state				
		Unpaired	Nearby	Connected (Owner)	Connected (non Owner)	Separated
Find My Network	Pairing control point	Available	Not available	Not available	Not available	Not available
Find My Network	Configuration control point	Not available	Not available	Available	Not available	Not available
Find My Network	Paired owner information control point	Not available	Not available	Available	Available	Available
Find My Network	Debug control point	Available in debug version only				
Firmware Update	All characteristics	Not available	Not available	Available	Not available	Not available
Accessory Non-Owner	All characteristics	Available	Not available	Not available	Available	Available

4.10.Serial number payload information

The payload parameters are defined in Table 4-42.

Table 4-42 Serial number payload information

Key	Size (Bytes)	NFC URL Format	Notes
b	1	ASCII Hex String	Battery status
bt	6	ASCII Hex String	Bluetooth MAC address
fv	4	ASCII Hex String	FW version, in little endian format

op	4	ASCII	For accessories that include NFC tag for serial number lookup, op value set to “tap”, otherwise it’s set to “bt”
e	141	ASCII Hex String	See Encrypted serial number payload). Available only when it’s in paired state.
pid	8	ASCII Hex String	Accessory product data
sr	16	ASCII	This is only available when it’s unpaired. Accessory serial number can be up to 16 bytes of uppercase alphanumeric characters (A-Z, 0-9). If the serial number is less than 16 bytes, trailing null padding shall be added.

4.10.1.Encrypted serial number payload

The encrypted payload (e) must be generated using the Apple server encryption key (Q_E), including the serial number, a counter, and a MAC computed using the KSN symmetric key. The counter (starting at 0) must monotonically increase every time after a NFC tap occurs (if the accessory uses NFC tag for serial number lookup) or every time after the Bluetooth serial number lookup control point is triggered.

Encrypted Payload: Serial Number || Counter || HMAC-SHA256(KSN, SerialNumber || Counter || op) || op

See ECIES Encryption for generating encrypted payload.

5. Advertisements

5.1. Bluetooth LE advertising

5.1.1. Payload for pairing

An accessory that is not Find My network paired shall advertise the Find My network service as a primary service when the user puts the accessory in pairing mode. The Bluetooth LE payload for pairing is defined in Table 5-1.

Table 5-1 Payload for Pairing state

Byte	Value	Description
0..5	MAC address	
6	0x17	Length of the Find My Service payload
7	0x16	16 bit UUID service data type
8-9	0xFD44	16 bit UUID for Find My network service
10 -17	Product data	See Product data
18-25	Accessory category	See Accessory Category for details Byte 0: Uint8 value of Accessory Category Byte 1-7: Reserved
26-29	Reserved	Reserved
30	Battery state	Battery state definition 0 = Full 1 = Medium 2 = Low 3 = Critically low

5.1.2. Payload for nearby state

After Find My network pairing, the accessory shall advertise the Find My network Bluetooth LE payload.

When the accessory is in the nearby state or connected to a paired owner device, the advertising payload format must be as defined in Table 5-2.

The nearby or separated state of the accessory determines the current key. The address type shall be set as a non-resolvable private address, or as a static device address.

The Find My network advertisement payload shall not contain other data types. An accessory must always advertise the Find My network payloads once every $T_{FMN\_ADV\_INTERVAL}$. The accessory may use another advertising instance to broadcast other data types and services.

Table 5-2 Payload for nearby state

Byte	Value	Description
0..5	MAC address	
6	0x07	Length of payload
7	0x16	Service Data AD type
8..9	0xFCB2	Unwanted Tracking Service UUID
10	0x01	Network (0x01 — Apple)
11	0x01	Near Owner (1 bit, least significant bit) + reserved (7 bits)
12	Bits 0–1: Reserved. Bit 2: Maintained Bits 3–4: Reserved Bits 5: 0b1 Bits 6–7: Battery state.	Maintained Set if owner connected within current key rotation period (15 minutes) Battery state definition 0 = Full 1 = Medium 2 = Low 3 = Critically low
13	Bits 0–1: Public key Bits 2–7: Reserved	Bits 6–7 of byte 0 of the primary key (P_i)

5.1.3. Payload for separated state

When the accessory is in the separated state, the advertising payload format must be as defined in Table 5-3.

Table 5-3 Payload for separated state

Byte	Value	Description
0..5	MAC address	
6	0x1E	Length of payload (30 bytes)
7	0x16	Service Data AD type
8..9	0xFCB2	Unwanted Tracking Service UUID
10	0x01	Network (0x01 — Apple)
11	0x00	Near Owner (1 bit, least significant bit) + reserved (7 bits)

Byte	Value	Description
12	Bits 0–1: Reserved. Bit 2: Maintained Bits 3–4: Reserved Bits 5: 0b1 Bits 6–7: Battery state.	Maintained Set if owner connected within current key rotation period (15 minutes) 0 = Full 1 = Medium 2 = Low 3 = Critically low
13 — 34	Separated public key	Bytes 6–27 of the Public Key, P_i or PW_j depending on accessory state. See Nearby to separated, Separated to separated, and After power cycle for possible separated state transitions.
35	Bits 0–1: Public key Bit 2: Motion detected Bits 3–7: Reserved	Bits 6–7 of byte 0 of the public key (P_i or PW_j)
36	Hint	Byte 5 of the Bluetooth address of the current primary key P_i

5.1.4. Advertisement in low battery state

If an accessory is unable to generate or rotate public keys due to low battery, it must stop advertising Find My network payload.

6. Pairing and Key Management

6.1. Overview

An accessory must be paired to an owner device before it can be locatable. An owner device will initiate the standard Bluetooth LE encryption before it accesses the Find My network services.

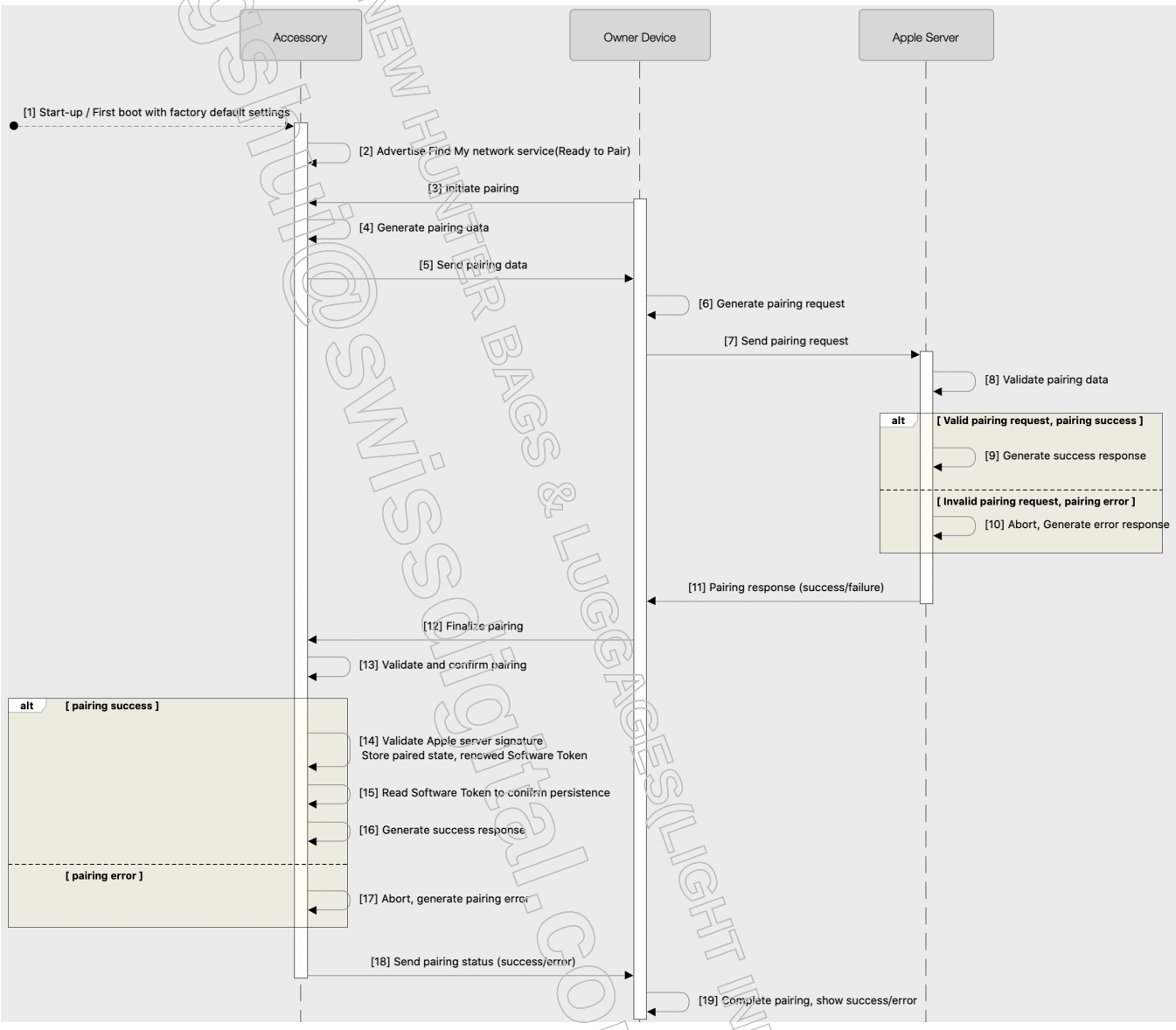

Figure 6-1: Accessory pairing flow

6.2. Pairing

Find My network pairing is initiated by the owner device using the pairing control point procedures. After an accessory pairs, it must not expose the Find My network pairing control point and it must respond to any of the pairing control point procedures with an invalid_command error message.

The accessory is associated with the Apple Account that is used to log into the owner device at the time of Find My network pairing.

An accessory will not be able to Find My network pair if it is paired to an owner device with a different Apple Account.

6.2.1. Pairing mode

The accessory must require explicit user intent to enable the Find My network pairing mode. When the user initiates the Find My network pairing mode, the accessory must advertise the Find My network service as a primary service. See section on pairing payload. The accessory must exit the pairing mode after 10 minutes.

6.2.2. Generate pairing data

Upon establishing standard Bluetooth LE encrypted pairing session, the accessory must generate collaborative commitment (C1) to start the pairing process and generate per pairing session encryption key seed (SeedK1). See Random generation for the generation of SeedK1. The accessory must regenerate SeedK1 for every new pairing session.

See Collaborative key generation for C1 details.

See Send pairing data pairing control point for details.

6.2.3. Send pairing data

The accessory must send encrypted payload generated using Apple server encryption key (Q_E).

The parameters listed in Table 6-1 are included in generating E2. See ECIES Encryption for E2 generation.

Table 6-1 Payload to generate E2

Key	Data type	Size (octets)	Description
SessionNonce	bytes	32	Nonce generated by Apple device
Software auth token	bytes	1024	Software authentication token that's vended by Apple for each accessory. If the token is less than 1024 bytes, trailing null padding shall be added.

Key	Data type	Size (octets)	Description
Software auth UUID	bytes	16	Accessory UUID that's associated with software auth. This shall be transmitted as big-endian byte order (that is, the most significant bytes are sent first).
Serial Number	ASCII	16	Accessory serial number
Product Data	bytes	8	Accessory product data
FW Version	bytes	4	Accessory firmware version
E1	bytes	113	Encrypted blob generated by owner device
SeedK1	bytes	32	Per pairing session seed for encryption key. See Generating pairing data for the generation of SeedK1

6.2.4. Finalize pairing

The owner device initiates the finalize pairing process to complete pairing. See Finalize pairing for details.

6.2.5. Validate and confirm pairing

The accessory must validate the Apple server signature (S2) using an Apple server signature verification key (Q_A) in order to finalize pairing.

The parameters listed in Table 6-2 are included in generating S2.

Table 6-2 Payload to generate signature message for S2 verification

Key	Data type	Size (octets)	Description
Software auth UUID	bytes	16	Accessory UUID that's associated with software token
SessionNonce	bytes	32	Nonce generated by owner device
SeedS	bytes	32	Unique server seed for each accessory that's paired
H1	bytes	32	Compute H1=SHA-256(C2)
E1	bytes	113	Encrypted blob generated by owner device
E3	bytes	1040	Encrypted software token that's vended by Apple server for each accessory

See Apple server public keys and ECDSA signature verification for signature verification key (Q_A) details.

In case of signature verification failure, the accessory must abort pairing. See Send Pairing Status for more details about success and error status.

If Apple server signature verification is successful, then the accessory must decrypt Apple server encrypted blob (E3) using the per pairing session symmetric AES 128-bit key K1 and the initialization vector IV1.

See Derivation of the pairing session key K1 and initialization vector IV1 for details on obtaining K1 and IV1. See AES-GCM decryption for E3 decryption details.

If S2 verification and E3 decryption are successful, then the accessory must store a new software token from E3 and generate a collaborative key (C3) as an acknowledgement to confirm pairing.

The accessory must always use the latest (renewed) software token for any subsequent operations that require authentication with Apple servers (for example, unpair).

See Collaborative key generation for C3 details. See Finalize pairing for E3 details.

6.2.6. Send pairing status

After successful pairing, the accessory must go into nearby state and send an acknowledgement to the owner device to confirm the pairing.

The accessory must initialize a unsigned 64-bit counter to 0. This counter is used along with the serial number in the NFC payload and Bluetooth Serial number lookup control point.

In case of pairing error, the accessory must abort pairing and send a pairing error code. For both success and error, the accessory must generate an encrypted blob (E4) and send it to the owner device.

The payload parameters listed in Table 6-3 are included in generating E4. See ECIES Encryption for E4 generation.

Table 6-3 Payload to generate E4

Key	Data type	Size (octets)	Description
Software auth UUID	bytes	16	Accessory UUID that's associated with software token
Serial Number	ASCII	16	Accessory serial number
SessionNonce	bytes	32	Nonce generated by the owner device
E1	bytes	113	Encrypted blob generated by the owner device
Software token	bytes	1024	Latest Software token
Status	UInt32	4	Success/failure status code

See Send pairing status for details.

6.3. Key management

6.3.1. Key definitions

As part of a successful pairing flow, the accessory and the owner device will collaboratively generate both of the following:

- A master public key, P
- Two symmetric keys, SKN and SKS

A derivative of the public key P will be broadcast over Bluetooth LE. Finder devices can use it to encrypt their current location and provide it to Apple servers for the accessory owner to download and decrypt.

Additionally, the accessory and the server generate a shared secret. The shared secret is used to derive a key and protects requests related to obtaining lost mode information:

- Secret shared with server: ServerSharedSecret
- Symmetric key for pairing session: K1
- Symmetric key for queries with serial number: KSN

6.3.2. Key sequences and rotation policy

The accessory must generate public key sequences with different key rotation intervals, referred to as primary and secondary keys.

- P and SKN are used to derive the primary key (P_i), which rotates every 15 minutes.
- P and SKS are used to derive the secondary key (PW_j), which rotates every 24 hours (that is, after every 96 iterations of primary key P_i).

6.3.3. Bluetooth advertisement key selection policy

6.3.3.1. After pairing

The accessory must use the primary key P_i (where i=1) as a Bluetooth LE advertisement and enters nearby state. See Payload for nearby state for details.

6.3.3.2. Nearby to nearby state transition

If at the end of period “i” the accessory is still in nearby state, it must use the next primary key P_{i+1} (where “i” is the last primary key index) as a Bluetooth LE advertisement. See Payload for nearby state for details.

6.3.3.3. Nearby to separated state transition

When the accessory switches to separated state, it must continue to use the current latched separated key P_i as a Bluetooth LE advertisement until the end of the current separated key period (4 a.m. local time). See Payload for separated state for details. See Latch separated key for details.

6.3.3.4. Separated to separated state transition

If at the end of the current separated key period (4 a.m. local time) the accessory is still in separated state, and it was previously advertising the last primary key P_i right after the state transition, it must compute j=i/96+1 and the next secondary key PW_j and use the latter as a Bluetooth LE advertisement.

See Payload for separated state for details.

6.3.3.5.Separated to connected / nearby state transition

When the accessory switches to nearby state, it must use the current primary key P_i as a Bluetooth LE advertisement. See Payload for nearby state for details.

6.3.3.6. After power cycle

The accessory must compute j=i/96+1 and the secondary key PW_j (where "i" is the current primary key index) and use the latter as a Bluetooth LE advertisement. See Payload for separated state for details.

6.3.4. Key schedule definitions

a || b denotes concatenation of the values a and b.

G is the base point of the NIST P-224 elliptic curve. See NIST SP 800-186, 3.2.1.2. P-224.

q is the order of the base point G. x(P) denotes the x coordinate of the elliptic curve point P.

ANSI-X9.63-KDF(Z, sharedInfo) denotes the KDF described by SEC1, 3.6.1 ANSI X9.63 Key Derivation Function. Z is the secret value (the input key material) and sharedInfo is data shared between the two parties. Hash() is the SHA-256 cryptographic hash function.

Random values and scalars must be generated using a cryptographically secure DRBG. See Operations.

6.3.4.1. Collaborative key generation

As part of the pairing flow, the owner device and the accessory must collaboratively generate a public key P and two symmetric keys, SKN and SKS.

1. The accessory generates two P-224 scalars s and r. (See Random scalar generation.) It computes the public point R = r · G and sends the value C1 = SHA-256(s || R), where len(C1) = 32 bytes, to the owner device. (See Send pairing data.)
2. The owner device generates two P-224 scalars s' and r'. (See Random scalar generation.) It computes S' = s' · G and R' = r' · G and sends C2 = S' || R', where len(C2) = 114 bytes, to the accessory. (See Finalize pairing.)
3. The accessory checks S' and R' and aborts if either is not a valid point on the curve. (See Elliptic curve point validation.) It computes the final public key P = S' + s · G and sends C3 = s || R, where len(C3) = 85 bytes, to the owner device. (See Send pairing status.)

4. The owner device aborts if s is not a valid P-224 scalar (see [Scalar validation]) or R is not a valid point on the curve (see [Elliptic curve point validation]) or if C1 ≠ SHA-256(s II R). It computes the final private key $d = s + s'$ (mod q) and the public key $P = d \cdot G$.
5. Both the owner device and the accessory compute the final symmetric keys SKN and SKS as the 64-byte output of ANSI-X9.63-KDF(x(P), RR), where $RR = r \cdot R'$ (for the accessory) or $RR = r' \cdot R$ (for the owner). SKN is the first 32 bytes and SKS is the last 32 bytes.

6.3.4.2. Derivation of primary and secondary keys

The accessory must derive primary and secondary keys from the public key P generated at pairing time. P itself must never be sent out and must be stored in a secure location.

For a given 15-minute period i:

1. Derive SKN_i = ANSI-X9.63-KDF(SKN_{i-1}, “update”), where SKN_0 is the SKN as agreed upon at pairing time.
2. Derive $AT_i = (u_i, v_i)$ = ANSI-X9.63-KDF(SKN_i, “diversify”) where $len(AT_i) = 72$ bytes and $len(u_i) = len(v_i) = 36$ bytes.
3. Reduce the 36-byte values u_i, v_i into valid P-224 scalars by computing the following:
   a. $u_i = u_i \pmod{q-1} + 1$
   b. $v_i = v_i \pmod{q-1} + 1$
4. Compute $P_i = u_i \cdot P + v_i \cdot G$.

Secondary keys are generated as shown above, using period j instead of i and SKS instead of SKN. The result will then be called PW_j instead of P_i.

6.3.4.3. Derivation of link encryption key LTK_i

The Find My network key generation algorithm generates LTKs, rotating every 15 minutes. The accessory shall use the LTK that corresponds to the current key period as the LTK to encrypt the link on connection to the owner device. A paired owner device also picks the same LTK to encrypt the link. If the device is not a paired Apple device or if the LTK results in a failed encryption, the accessory must disconnect.

The accessory must derive a new link encryption key LTK_i for every 15-minute period i. If the paired owner device is nearby, it can use this key to establish a Bluetooth connection and encrypt the link.

For a given 15-minute period i:

1. Derive the symmetric key SKN_i = ANSI-X9.63-KDF(SKN_{i-1}, “update”), where SKN_0 is the symmetric key SKN as agreed upon at pairing time.
2. Derive the Intermediate key IK_i = ANSI-X9.63-KDF(SKN_i, “intermediate”), where $len(IK_i) = 32$ bytes.
3. Derive the Link Encryption key LTK_i = ANSI-X9.63-KDF(IK_i, “connect”), where $len(LTK_i) = 16$ bytes.

6.3.4.4. Derivation of ServerSharedSecret

Upon successful pairing, the accessory must generate and retain ServerSharedSecret, where ServerSharedSecret is a 32-byte shared secret:

ServerSharedSecret = ANSI-X9.63-KDF(SeedS || SeedK1, “ServerSharedSecret”)

6.3.4.5. Derivation of pairing session key K1 and initialization vector IV1

The 16-byte symmetric key K1 and the 16-byte initialization vector IV1 must be generated as follows:

K1 || IV1 = ANSI-X9.63-KDF(ServerSharedSecret, “PairingSession”)

Where K1 is the first 16 bytes and IV1 the last 16 bytes of the KDF output.

6.3.4.6.Derivation of the serial number protection key

To generate the NFC tap payload, KSN must be generated as follows, where KSN is a 32-byte symmetric key:

KSN = ANSI-X9.63-KDF(ServerSharedSecret, “SerialNumberProtection”)

6.4. Unpair procedure

Unpair action is initiated by the paired owner device to delete Find My network data and to remove Find My network pairing on the accessory. See Unpair for the unpair opcode.

The accessory shall complete the unpair procedure only after a Bluetooth LE disconnect is initiated by the paired owner device. A Bluetooth LE connection timeout or other failures after an unpair action is initiated does not complete the unpair action. See Factory reset for details on resetting the accessory.

7. Unwanted Tracking Detection

7.1. Overview

During separated state, sound playback from the accessory is designed to bring awareness to the person with whom it's detected. Accessories that support motion-triggered UT sound alerts (see Product-specific requirements) must implement the requirements from this chapter.

7.2. Hardware

7.2.1. Motion detector

The accessory must include a motion detector that can detect accessory motion reliably (for example, an accelerometer). If the accessory includes an accelerometer, it must be configured to detect an orientation change of ±10° along any two axes of the accessory.

7.2.2. Sound maker

The accessory must include a sound maker (for example, a speaker) to play sound when motion is detected in separated state.

It must also play sound when a non-owner tries to locate the accessory by initiating a play sound command from a non-owner device when the accessory is in range and connectable through Bluetooth LE. See Play sound—non-owner control point.

The sound maker must emit a sound with minimum 60 phon peak loudness as defined by *ISO 532-1:2017*. The loudness must be measured in free acoustic space substantially free of obstacles that would affect the pressure measurement. The loudness must be measured by a calibrated (to the Pascal) free field microphone 25 cm from the accessory suspended in free space.

7.3. Implementation

After $T_{SEPARATED\_UT\_TIMEOUT}$ in separated state, the accessory must enable the motion detector (for example, accelerometer) to detect any motion within $T_{SEPARATED\_UT\_SAMPLING\_RATE1}$.

If motion is not detected within the $T_{SEPARATED\_UT\_SAMPLING\_RATE1}$ period, the accessory must stay in this state until it exits separated state.

If motion is detected within the $T_{SEPARATED\_UT\_SAMPLING\_RATE1}$ the accessory must play a sound. After first motion is detected, the movement detection period is decreased to $T_{SEPARATED\_UT\_SAMPLING\_RATE2}$. The

accessory must continue to play a sound for every detected motion. The accessory shall disable the motion detector for $T_{SEPARATED\_UT\_BACKOFF}$ under either of the following conditions:

- Motion has been detected for 20 seconds at $T_{SEPARATED\_UT\_SAMPLING\_RATE2}$ periods.
- Play sound has been attempted 10 times.

If the accessory is still in separated state at the end of $T_{SEPARATED\_UT\_BACKOFF}$, the UT behavior must restart.

A Bluetooth LE connection from an owner Apple device must reset the separated behavior and transition the accessory to connected state.

8. NFC Requirements

8.1. Overview

Accessories that include NFC (see Serial number lookup) must support the requirements from this chapter.

8.2. Hardware

These are the hardware requirements for accessories that include NFC:

- The accessory must use a programmable NFC tag.
- NFC tags must use the NFC Data Exchange Format (NDEF) as defined by *NFC Forum™ in NDEF 1.0 NFCForum-TS-NDEF 1.0.*
- An NDEF message is defined as a group of individual NDEF records as defined by *NFC Forum™ in NFC Record Type Definition (RTD) RTD 1.0 NFCForum-TS-RTD 1.0.*
- The Find My network payload for NFC tags must use NDEF URI Record Type Definition as defined by *NFC Forum™ in URI Record Type Definition RTD-URI 1.0 NFCForum-TS-RTD URI 1.0*.
- The minimum payload that must be supported is 30 bytes.
- NFC tag types must be type 2 or greater.
- The NFC tag should not be scannable when the Find My network-enabled accessory is still in the packaging.
- The Find My network payload must be scannable when holding the top of the iOS controller near the center of the NFC tag on the accessory. Recommended NFC tag performance guidelines are defined by *NFC Forum™ in Tag Performance Requirements Document.*
- The NFC on the accessory must be configured as a NFC tag.

8.3. Implementation

Accessories must advertise the following payload over NFC.

- Unpaired accessories advertise the following payload:
  https://found.apple.com/accessory?pid=%04x&b=%02x&fv=%08x&bt=%s&sr=%s
- Paired accessories advertise the following payload:
  https://found.apple.com/accessory?pid=%04x&b=%02x&fv=%08x&e=%s&op=%s

Please see Serial number payload information for payload parameters.

9. Timers and Constants

9.1. Overview

Table 9-1 defines the timers and constants used by the Find My network protocol.

Table 9-1 Timers and constants

Timer Name	Value	Description
TSEPARATED_UT_SAMPLING_RATE2	0.5 seconds	Motion detector sampling rate when movement is detected in separated state.
TRECONNECT_ADV_INTERVAL	30 ms	Advertising interval for reconnection attempt when a persistent connection is lost
TRECONNECT_ATTEMPT_TIMEOUT	3 seconds	Advertising duration for reconnection attempt
TFMN_ADV_INTERVAL	2 seconds	Find My network Bluetooth LE Advertising Interval.
TSEPARATED_UT_SAMPLING_RATE1	10 seconds	Sampling rate when Motion detector is enabled in separated state.
TFMN_SEPARATED_SN_LOOKUP_INTERVAL	5 minutes	Sampling rate when serial number lookup is enabled in separated state.
TNEARBY	15 minutes	Default value. Configured by the owner device on connection.
TSEPARATED_UT_BACKOFF	6 hours	Period to disable motion detector if accessory is in separated state.
TSEPARATED_UT_TIMEOUT	random value between 8-24 hours chosen from a uniform distribution	Time span in separated state before enabling motion detector.

10. Firmware Update

10.1.Overview

The unified accessory restore protocol (UARP) should be used to update firmware on an accessory. This protocol uses the Firmware update service to transfer UARP messages between the accessory and the owner device.

Details about the unified accessory restore protocol will be found in the *Unified Accessory Restore Protocol Development Guide*.

11. Accessory Categories

Accessory manufacturer's must pick an accessory category that closest resembles their physical product. The accessory categories outlined here are used for presentation purpose by the Find My app. If none of the accessory categories provided in this list match the physical product, Other must be chosen.

Table 11-1 Accessory Categories

Accessory Category	Value
Finder	1
Other	128
Luggage	129
Backpack	130
Jacket	131
Coat	132
Shoes	133
Bike	134
Scooter	135
Stroller	136
Wheelchair	137
Boat	138
Helmet	139
Skateboard	140
Skis	141
Snowboard	142
Surfboard	143
Camera	144
Laptop	145
Watch	146
Flash drive	147
Drone	148

Accessory Category	Value
Headphones	149
Earphones	150
Inhaler	151
Sunglasses	152
Handbag	153
Wallet	154
Umbrella	155
Water bottle	156
Tools or tool box	157
Keys	158
Smart case	159
Remote	160
Hat	161
Motorbike	162
Consumer electronic device	163
Apparel	164
Transportation device	165
Sports equipment	166
Personal item	167
Luggage Tag	168
Reserved for future use	2-127, 169+

12. App integration

12.1.Overview

An accessory manufacturer may optionally provide an iOS accessory app to allow users to setup, configure and use their accessories. This chapter defines features an accessory app may include to help integrate an iOS accessory app with the Apple Find My app and Find My network.

12.2.General

Universal Links allow an iOS accessory app to interact with the Apple Find My app to allow basic operations with Find My network accessories. By leveraging Universal Links, an accessory app has the ability to trigger a pairing request in the Find My app, jump directly to the detail card of the accessory, including showing updated offline locations, and also allow linking to the Apple Find My app to remove the paired accessory from the Find My network. An accessory app should programmatically build the URL's to link to the Apple Find My app using the formats specified below and then call openURL from your application to launch the Apple Find My app. See Universal Links Apple Developer documentation for more details.

12.3.Supported URLs

12.3.1.Setup item

12.3.1.1.Supported platform

iOS 14.5 or later

12.3.1.2.Details

An accessory app should instruct users that after pairing they should return to the app for continuation of any additional setup procedures. At this point the accessory app should query the accessory to determine Find My pairing status.

Example URL:

http://findmy.apple.com/item?action=setup

Table 12-1 Setup item

Property	Example	Description
Setup Link Base URL (required)	http://findmy.apple.com/item?action=setup	Base URL for Setup Universal Link

12.3.2.Select item

12.3.2.1.Supported platform

iOS 14.5 and macOS 11.5 or later

12.3.2.2.Details

An accessory app can directly deep link to your accessory by passing the serial number and manufacturer name to the Apple Find My app. The Apple Find My app will switch to the appropriate items tab, select the item, and present the details pane as well as fetch updated offline location for the accessory.

Example URL:

http://findmy.apple.com/item?serial=123456789&manufacturer=My%20Company

Table 12-2 Select item

Property	Example	Description
Details Base URL (required)	http://findmy.apple.com/item	Base URL for details Universal Link
Serial (required)	serial=123456789	Serial number of the item you are selecting.
Manufacturer (required)	manufacturer=My%20Company	Manufacturer should match the manufacturer string shown in the details pane of the Find My app.

12.3.3.Remove item

12.3.3.1.Supported platform

iOS 14.5 or later

12.3.3.2.Details

An accessory app can link directly to Apple Find My app and bring up the removal sheet, which will unpair the accessory from Apple Find My app and Find My network. Licensee app should verify the

removal was successful by querying the accessory directly when users make their way back to your app.

Example URL:

http://findmy.apple.com/item?action=remove&serial=123456789&manufacturer=My%20Company

Table 12-3 Remove item

Property	Example	Description
Removal Base (required)	http://findmy.apple.com/item?action=remove	Base URL for removal Universal Link
Serial (required)	serial=123456789	Serial number of the item you are selecting to remove.
Manufacturer (required)	manufacturer=My%20Company	Manufacturer should match the manufacturer string showed in the details pane of the Find My app.

13. Revision History

This chapter describes the changes to *Find My Network Accessory Specification* from the previous revision.

- Updated Section 1 to Section 8
- Updated 3.2 General
- Updated Section 3.3.1 Bluetooth
- Updated 3.3.4 Serial number lookup
- Added clarifications on Accessory Category bytes, Table 4-5 and Table 5-1

