# OpenCable™ Specifications

# Cable Profile for the ZigBee® RF4CE Remote Control Specification

# OC-SP-RF4CE-I01-120924

## ISSUED

# DISCLAIMER

This document is published by Cable Television Laboratories, Inc. ("CableLabs®").

CableLabs reserves the right to revise this document for any reason including, but not limited to, changes in laws, regulations, or standards promulgated by various agencies; technological advances; or changes in equipment design, manufacturing techniques, or operating procedures described, or referred to, herein. CableLabs makes no representation or warranty, express or implied, with respect to the completeness, accuracy, or utility of the document or any information or opinion contained in the report. Any use or reliance on the information or opinion is at the risk of the user, and CableLabs shall not be liable for any damage or injury incurred by any person arising out of the completeness, accuracy, or utility of any information or opinion contained in the document.

This document is not to be construed to suggest that any affiliated company modify or change any of its products or procedures, nor does this document represent a commitment by CableLabs or any cable member to purchase any product whether or not it meets the described characteristics. Nothing contained herein shall be construed to confer any license or right to any intellectual property, whether or not the use of any information herein necessarily utilizes such intellectual property. This document is not to be construed as an endorsement of any product or company or as the adoption or promulgation of any guidelines, standards, or recommendations.

# Document Status Sheet

| | | | | |
|---|---|---|---|---|
| **Document Control Number:** | OC-SP-RF4CE-I01-120924 | | | |
| **Document Title:** | Cable Profile for the ZigBee® RF4CE Remote Control Specification | | | |
| **Revision History:** | I01 – Released 9/24/12 | | | |
| **Date:** | September 24, 2012 | | | |
| **Status:** | ~~Work in Progress~~ | ~~Draft~~ | **Issued** | ~~Closed~~ |
| **Distribution Restrictions:** | ~~Author Only~~ | ~~CL/Member~~ | ~~CL/ Member/ Vendor~~ | **Public** |

## Key to Document Status Codes

| | |
|---|---|
| **Work in Progress** | An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration. |
| **Draft** | A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process. |
| **Issued** | A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing. |
| **Closed** | A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs. |

## Trademarks

# Contents

# Figures



# Tables

# 1 SCOPE

## 1.1 Introduction and Purpose

The RF4CE MSO Profile is a profile for RF4CE [RF4CE], and is specifically designed to facilitate remote control of a target by a controller in a cable set-top environment. The RF4CE MSO Profile is based on the ZigBee Remote Control (ZRC) profile [RF4CE ZRC] with additions and changes to make it suitable for a cable user environment. These changes include button-less pairing, data storage, and access on the target for the remote. Buttonless pairing will allow cable users to pair a remote without having to press a physical button on the target device, and remote storage will facilitate stateless operation of the remote and IR Setup functionality.

## 1.2 Purpose of document

This document defines detailed requirements for the RF4CE MSO Profile. It is intended to provide specific guidance for all required cable environment functionality for an RF4CE remote control.

## 1.3 Organization of document

The document is organized into five sections as follows:

Sections 1 - 4 present basic information, including references and acronyms.

Section 5 is an overview.

Section 6 provides specifications and requirements for the RF4CE MSO Profile.

## 1.4 Scope of document

This document specifies the required RF4CE MSO Profile functions and operations to enable control of RF-capable cable devices.

This document does not specify requirements for head-end, back-office systems, or any of the systems or servers used to source video services.

## 1.5 Requirements

The following words are used throughout this document to define the significance of particular requirements.

| | |
|---|---|
| SHALL | This word means that the item is an absolute requirement of this specification. |
| SHALL NOT | This phrase means that the item is an absolute prohibition of this specification. |
| SHOULD | This word means that valid reasons may exist in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course. |
| SHOULD NOT | This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label. |
| MAY | This word, or the adjective means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item. |

# 2 REFERENCES

## 2.1 Normative References

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references.

All references are subject to revision, and parties to agreement based on this specification are encouraged to investigate the possibility of applying the most recent editions of the documents listed below.

| | |
|---|---|
| [HDMI 1.4] | High-Definition Multimedia Interface Specification, HDMI Licensing, LLC; Version 1.4, June 5, 2009. |
| [IEEE802.15.4] | IEEE Standard for Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2003. |
| [RF4CE] | ZigBee RF4CE Specification, Version 1.01; ZigBee Document 094945r00ZB, January, 2010. |
| [RF4CE CODE] | ZigBee Manufacturer Code Database Specification, ZigBee Document 053874r15ZB_CSG-Manufacturer-Code-Database.pdf, March 15th, 2010. |
| [RF4CE ID] | ZigBee RF4CE Profile ID List, ZigBee Alliance, March 17th, 2009. |
| [RF4CE TYPE] | ZigBee RF4CE Device Type List, ZigBee Alliance, March 17th, 2009. |
| [RF4CE ZRC] | Consumer Electronics Remote Control Profile Specification, Version 1.10; ZigBee Document 094946r00ZB. |

## 2.2 Informative References

This specification uses the following informative references.

| | |
|---|---|
| [HDMI 1.3] | High-Definition Multimedia Interface Specification, HDMI Licensing, LLC; Version 1.3a, November 10, 2006. |

## 2.3 Reference Acquisition

- Institute of Electrical and Electronics Engineers (IEEE), +1 800 422 4633 (USA and Canada); http://www.ieee.org
- ZigBee Alliance Inc., 2400 Camino Ramon, Suite 375, San Ramon, CA 94583; http://zigbee.org

# 3 TERMS AND DEFINITIONS

This specification uses the following terms:

| | |
|---|---|
| **Consumer Electronics Control** | A feature that allows the user to command and control multiple CEC-enabled, HDMI-connected devices. |
| **Digital Transport Adapter** | A cable receiver that enables conversion of audio/video signals from digital MPEG-2 based transport into analog or digital display. |
| **High Definition Universal Digital Transport Adapter** | A DTA that delivers converted A/V signals in HD format, and which can operate in a number of content protection modes, including Motorola Copy Protection System (CPS), Cisco Systems' Simple Content Protection, as well as DVB-compliant content protection systems. |
| **Infra-red Database** | A comprehensive list of model-specific TV IR codes which is stored in an RF target device's firmware; specific codesets are transmitted to the controller to enable TV remote control. |
| **Network Information Base** | A collection of RF4CE network information used by compatible devices during the network discovery process. |
| **Network Layer Management Entity** | A portion of the ZigBee stack that enables and manages device communication, including device initialization, message routing and network discovery. |
| **Radio Frequency for Consumer Electronics** | A consumer electronics consortium working jointly with the ZigBee Alliance to deliver a standardized specification for radio frequency-based remote controls. |
| **ZigBee** | A low-cost, low-power, wireless mesh networking standard. |
| **ZigBee Remote Control** | A remote control supporting the standard RF4CE communication profile. |

# 4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

| | |
|---|---|
| **CEC** | Consumer Electronics Control |
| **DTA** | Digital Transport Adapter |
| **HD-uDTA** | High Definition Universal Digital Transport Adapter |
| **IEEE** | Institute of Electrical and Electronic Engineers |
| **IR** | Infrared |
| **IRDB** | Infra-red Database |
| **LQI** | Link Quality Indicator |
| **MSO** | Multiple System Operators |
| **NIB** | Network Information Base |
| **NLME** | Network Layer Management Entity |
| **RC** | Remote Control |
| **RF** | Radio Frequency |
| **RF4CE** | Radio Frequency for Consumer Electronics |
| **RIB** | Remote Information Base |
| **STB** | Set Top Box |
| **ZRC** | ZigBee Remote Control |

# 5 OVERVIEW

The RF4CE MSO Profile, an extension of the RF4CE protocol, will be implemented by cable RF4CE remote controls. These remote controls can then control any cable device that supports the RF4CE MSO Profile. These cable devices include, but are not limited to, STBs and HD-uDTAs. The RF4CE MSO Profile is based on the RF4CE ZRC [RF4CE ZRC] profile with added support for button-less pairing and retrieval of data from the RF target device by the remote.

By using the dynamic IR code database implemented with this remote control system, customers will be able to set up TV controls interactively, guided by on-screen instructions when operating STB or HD-uDTA devices which support the RF4CE MSO Profile.

In addition, a target device can be connected via HDMI [HDMI 1.4] to TVs and other devices that support Consumer Electronics Control (CEC) commands. TVs connected in this way can be controlled through the RF target device via the remote control, as specified by the target device's hardware requirements. For TVs not connected by HDMI, or not compatible with CEC, IR signaling will be used for remote control.

# 6 RF4CE MSO PROFILE SPECIFICATION

The following sections identify the RF4CE MSO Profile Protocol support requirements.

## 6.1 General MSO Command Frame Format

The general MSO command frame SHALL be formatted as illustrated in Table 1.

***Table 1 - General MSO Command Frame Format***

| 1 Octet | Variable |
|---|---|
| Frame Control | Command Payload |
| MSO header | MSO payload |

### 6.1.1 Frame Control Field

The frame control field SHALL be one (1) octet in length.

The frame control field SHALL contain information defining the MSO command code.

The frame control field SHALL be formatted as illustrated in Table 2.

***Table 2 - General MSO Command Frame Format***

| 1 Octet |
|---|
| MSO Command Code |
| Frame Control |

#### *6.1.1.1 MSO Command Code*

The MSO command code sub-field SHALL be eight (8) bits in length and SHALL be set to one of the non-reserved values listed Table 3. Each of these commands is specified in detail in Section 6.2.

***Table 3 - Values of the MSO Frame Control Field***

| MSO command code | Description |
|---|---|
| 0x00 | Reserved |
| 0x01 | User control pressed |
| 0x02 | User control repeated |
| 0x03 | User control released |
| 0x04 – 0x1F | Reserved |
| 0x20 | Check validation request |
| 0x21 | Check validation response |
| 0x22 | Set attribute request |
| 0x23 | Set attribute response |
| 0x24 | Get attribute request |
| 0x25 | Get attribute response |
| 0x26 – 0xff | Reserved[1] |

[1]Other vendor specific frames can be added if required by the application.

### 6.1.2 Command Payload Field

The command payload field SHALL be of variable length and contain information specific to individual MSO commands.

## 6.2 MSO Command Frames

### 6.2.1 User Control Pressed Command Frame

The user control pressed command frame SHALL be supported as specified in [RF4CE ZRC].

### 6.2.2 User Control Repeated Command Frame

The user control repeated command frame SHALL be supported as specified in [RF4CE ZRC].

### 6.2.3 User Control Released Command Frame

The user control released command frame SHALL be supported as specified in [RF4CE ZRC].

### 6.2.4 Check Validation Request Command Frame

The check validation request command frame allows a remote node to check the result of a validation procedure on a target device.

The check validation request command frame SHALL be formatted as illustrated in Table 4.

***Table 4 - Format of the Check Validation Request Command Frame***

| 1 Octet | Variable |
|---|---|
| Frame Control | Check Validation Control |
| MSO header | MSO payload |

The check validation request command SHALL be directed to a target node that is already in the pairing table.

#### *6.2.4.1.1 Check Validation Control*

The check validation control field SHALL be one (1) octet in length.

The check validation control field SHALL contain information that is used in the validation procedure as specified in Section 6.5.6.

The check validation control field SHALL be formatted as illustrated in Table 5.

***Table 5 - Format of the Check Validation Control Field***

| Bit 0 | Bits 1-7 |
|---|---|
| Request Automatic Validation | Reserved |
| Check Validation Control | |

##### 6.2.4.1.1.1 Request Automatic Validation

The Request Automatic Validation sub-field SHALL be one (1) bit in length.

If the bit is set to '1', it SHALL indicate that the controller requests to be validated automatically.

#### *6.2.4.1.2 Check Validation Response Command Frame*

The check validation response command frame allows a target node to transfer the result of the validation procedure (requested by a remote node using the check validation request) to a remote node.

The check validation response command SHALL be directed to a remote node that is already in the pairing table.

The check validation response command frame SHALL be formatted as illustrated in Table 6.

***Table 6 - Format of the Check Validation Request Command Frame***

| 1 Octet | Variable |
|---|---|
| Frame Control | Check Validation Status |
| MSO header | MSO payload |

#### *6.2.4.2 Check Validation Status*

The check validation status field SHALL be one (1) octet in length.

The check validation status field SHALL contain the result of the validation procedure as specified Section 6.5.6.

Check Validation status values SHALL be as specified in Table 7.

***Table 7 - Check Validation Status***

| Name | Value | Generated by | Description |
|---|---|---|---|
| Success | 0x0 | Application | The validation is successful. |
| Pending | 0xc0 | Application | The validation is still in progress. |
| Time Out | 0xc1 | Profile | The validation timed out, and the binding procedure SHOULD continue with other devices in the list. |
| Collision | 0xc2 | Profile | The validation was terminated at the target side, as more than one controller tried to pair. |
| Failure | 0xc3 | Application | The validation failed, and the binding procedure SHOULD continue with other devices in the list. |
| Abort | 0xc4 | Application | The validation is aborted, and the binding procedure SHOULD continue with other devices in the list. |
| Full Abort | 0xc5 | Application | The validation is aborted, and the binding procedure SHOULD NOT continue with other devices in the list. |

### 6.2.5 Set Attribute Request Command Frame

The set attribute request command frame SHALL allow a remote node to store data in the Remote Information Base (RIB) of a target.

The set attribute request command SHALL be directed to a target node that is already in the pairing table.

The set attribute request command frame SHALL be formatted as illustrated in Table 8.

***Table 8 - Format of the Set Attribute Request Command Frame***

| 1 Octet | 1 Octet | 1 Octet | 1 Octet | Variable |
|---|---|---|---|---|
| Frame Control | Attribute Identifier | Index | Value Length | Value |
| MSO header | MSO payload | | | |

#### *6.2.5.1 Attribute Identifier*

The Attribute Identifier field SHALL be one (1) octet in length.

The Attribute Identifier field SHALL contain the identifier of the RIB attribute that the remote node wants to store, as specified in Table 23.

#### *6.2.5.2 Index*

The Index field SHALL be one (1) octet in length.

If the RIB attribute is represented as a vector, this field SHALL contain the index of the element that the remote node wants to store, as specified in the description of the different attributes in Section 6.5.8.1.

If the RIB attribute is represented as a single element, this field SHALL be ignored.

#### *6.2.5.3 Value Length*

The Value Length field SHALL be one (1) octet in length.

The Value Length field SHALL contain the length in octets of the Value field.

The maximal value of this field SHALL be *aplcMaxRIBAttributeSize*.

#### *6.2.5.4 Value*

The Value field SHALL have a variable length as indicated by the Value Length field.

The Value field SHALL contain the value of the attribute element that the remote node wants to store.

### 6.2.6 Set Attribute Response Command Frame

The set attribute response command frame SHALL allow a target node to transfer the result of a set attribute procedure (requested by a remote node using the set attribute request) to a remote node.

The set attribute response command SHALL be directed to a remote node that is already in the pairing table.

The set attribute response command frame SHALL be formatted as illustrated in Table 9.

***Table 9 - Format of the Set Attribute Response Command Frame***

| 1 Octet | 1 Octet | 1 Octet | 1 Octet |
|---|---|---|---|
| Frame Control | Attribute Identifier | Index | Status |
| MSO header | MSO payload | | |

#### *6.2.6.1 Attribute Identifier*

The Attribute Identifier field SHALL be one (1) octet in length.

The Attribute Identifier SHALL contain the identifier of the RIB attribute that the target attempted to store, as specified in Table 23.

#### *6.2.6.2 Index*

The Index field SHALL be one (1) octet in length.

If the RIB attribute is represented as a vector, this field SHALL contain the index of the element that the target attempted to store, as specified in the description of the different attributes in Section 6.5.8.1.

If the RIB attribute is represented as a single element, this field SHALL be ignored.

#### *6.2.6.3 Status*

The Status field SHALL be one (1) octet in length.

The Status field SHALL contain the result of the set attribute procedure on the target, as specified in Section 6.5.8.1.

### 6.2.7 Get Attribute Request Command Frame

The get attribute request command frame allows a remote node to retrieve data from the Remote Information Base (RIB) of the target.

The get attribute request command SHALL be directed to a target node that is already in the pairing table.

The get attribute request command frame SHALL be formatted as illustrated in Table 10.

***Table 10 - Format of the Get Attribute Request Command Frame***

| 1 Octet | 1 Octet | 1 Octet | 1 Octet |
|---|---|---|---|
| Frame Control | Attribute Identifier | Index | Value Length |
| MSO header | MSO payload | | |

#### *6.2.7.1 Attribute Identifier*

The attribute identifier field SHALL be one (1) octet in length.

The attribute identifier SHALL contain the identifier of the RIB attribute that the remote node wants to retrieve, as specified in Table 23.

#### *6.2.7.2 Index*

The Index field SHALL be one (1) octet in length.

If the RIB attribute is represented as a vector, this field SHALL contain the index of the element that the remote node wants to retrieve, as specified in the description of the different attributes in Section 6.5.8.1.

If the RIB attribute is represented as a single element, this field SHALL be ignored.

#### *6.2.7.3 Value Length*

The Value Length field SHALL be one (1) octet in length.

The Value Length field SHALL contain the length in octets of the requested attribute field.

The maximal value of this field SHALL be *aplcMaxRIBAttributeSize*.

### 6.2.8 Get Attribute Response Command Frame

The get attribute response command frame allows a target node to transfer the result of a get attribute procedure (requested by a remote node using the get attribute request) to a remote node, together with the retrieved data.

The get attribute response command SHALL be directed to a remote node that is already in the pairing table.

The get attribute response command frame SHALL be formatted as illustrated in Table 11.

***Table 11 - Format of the Get Attribute Response Command Frame***

| 1 Octet | 1 Octet | 1 Octet | 1 Octet | 1 Octet | Variable |
|---|---|---|---|---|---|
| Frame Control | Attribute Identifier | Index | Status | Value Length | Value |
| MSO header | MSO payload | | | | |

#### 6.2.8.1 Attribute Identifier

The Attribute Identifier field SHALL be one (1) octet in length.

The Attribute Identifier field SHALL contain the identifier of the RIB attribute that was retrieved, as specified in Table 23.

#### 6.2.8.2 Index

The Index field SHALL be one (1) octet in length.

If the RIB attribute is represented as a vector, this field SHALL contain the index of the element that was retrieved, as specified in the description of the different attributes in Section 6.5.8.1.

If the RIB attribute is represented as a single element, this field SHALL be ignored.

#### 6.2.8.3 Status

The Status field SHALL be one (1) octet in length.

The Status field SHALL contain the result of the get attribute procedure on the target, as specified in Section 6.5.8.9.

#### 6.2.8.4 Value Length

The Value Length field SHALL be one (1) octet in length.

The Value Length field SHALL contain the length in octets of the Value field.

The maximal value of this field SHALL be *aplcMaxRIBAttributeSize*, as specified in Table 12.

#### 6.2.8.5 Value

The Value field has a variable length as indicated by the Value Length field.

The Value field SHALL contain the value of the retrieved attribute element.

## 6.3 RF4CE MSO Profile Constants and Attributes

### 6.3.1 RF4CE MSO Profile Constants

The constants that define the characteristics of the RF4CE MSO Profile SHALL be as specified in this section.

These constants have impact on both controller and target devices, and SHALL be defined as specified in Table 12.

***Table 12 -Common RF4CE MSO Profile Constants***

| Constant | Description | Default Value |
|---|---|---|
| aplcMaxKeyRepeatInterval | The maximum time between consecutive user control repeated command frame transmission. | 120 ms |
| aplcMaxRIBAttributeSize | The maximum size in octets of the elements of the attributes in the RIB. At the same time, the maximum size in octets of the Value field in the set attribute request and get attribute response command frames. | 92 |
| aplcResponseIdleTime | The time a device SHALL wait after the successful transmission of a request command frame, before enabling its receiver to receive a response command frame | 50 ms |
| aplcBlackOutTime | The time at the start of the validation procedure during which packets SHALL NOT be transmitted. | 100 ms |
| aplcMinKeyExchangeTransferCount | The minimum value of the KeyExTransferCount parameter passed to the pair request primitive during the push button pairing procedure. | 3 |

### 6.3.2 RF4CE MSO Profile Attributes

The RF4CE MSO Profile attributes SHALL determine how the RF4CE MSO Profile operates.

These attributes SHALL be defined as specified in Table 13. Whether these attributes are set at compile time or at run time is implementation-specific.

***Table 13 - RF4CE MSO Profile Attributes***

| Attribute | Applicable to | Type | Range | Description | Default |
|---|---|---|---|---|---|
| aplKeyRepeatInterval[2] | Controller | Integer | 0 – aplcMax KeyRepeat Interval | The interval in ms at which user command repeat frames will be transmitted for repeatable keys. | 0.5 * aplcMax KeyRepeatInterval |
| aplKeyRepeatWaitTime[3] | Target | Integer | ≥aplcMax KeyRepeat Interval | The duration in ms that a recipient of a user control repeated command frame waits before terminating a repeated operation. | aplcMax KeyRepeatInterval |
| aplResponseWaitTime | Controller | Integer | 0x000000 – 0xffffff | The maximum time in symbols that a device SHALL wait (after the *aplcResponseIdleWaitTime* expired) to receive a response command frame following a request command frame. | 0x186a (100 ms)* |
| aplMaxPairingCandidates | Controller | Integer | 0x00 – nwkcMax NodeDesc ListSize | The maximum number of pairing candidates selected from the NLME-DISCOVERY.response node descriptor list. | 3 |
| aplLinkLostWaitTime | Controller | Integer | (aplcRe-sponseIdleTime + aplRe-sponseWaitTime) - 0xffff | The maximum time in ms that a device can stay in the validation procedure without receiving the responses corresponding to its requests.<br>[Can be updated by RIB procedure at the start of the validation procedure.] | 0x02710 (10000 ms) |
| aplAutoCheckValidationPeriod | Controller | Integer | (aplcRe-sponseIdleTime + aplRe-sponseWaitTime) - 0xffff | The time period in ms between the regular check validation requests that a controller transmits in the validation procedure.<br>[Can be updated by RIB procedure at the start of the validation.] | 0x01f4 (500 ms) |
| aplValidationWaitTime | Target | Integer | 0x0000 – 0xffff (0x0 to be interpreted as ∞) | The maximum time in ms that a device can stay in the validation procedure. | 0x0 (∞ ms) for buttonless validation<br>0x7530 (30000 ms) for button based validation |
| aplValidationInitialWatchdogTime | Target | Integer | 0x0000 – 0xffff (0x0 to be | The maximum time in ms that a device can stay in the | 0x1f40 (8000 ms) for |

[2]See [RF4CE ZRC]
[3]See [RF4CE ZRC]

| Attribute | Applicable to | Type | Range | Description | Default |
|---|---|---|---|---|---|
| | | | interpreted as ∞) | validation procedure, without receiving a first validation watchdog kick. | buttonless validation 0x0 (∞ ms) for button based validation |
| aplUserString | Target and controller | Character String | 9 characters | The user-defined character string to carry application-related information. | Empty String |
| aplKeyExchangeTransferCount | Controller | Integer | aplcMinKeyExchangeTransferCount – 0xff | The value of the KeyExTransferCount parameter passed to the pair request primitive during the temporary pairing procedure. | 0x4 |
| *Some timing values within this specification are specified in terms of MAC symbols. One MAC symbol is equal to 16µs. Where appropriate, absolute time values are presented in both MAC symbols and actual time in parentheses. | | | | | |

## 6.4 Functional Description

### 6.4.1 Initialization

A node operating according to the RF4CE MSO Profile SHALL configure the NIB attributes specified in Table 14.

All other NIB attributes SHALL be set to their default values.

***Table 14 - RF4CE MSO Profile Initial NIB Attribute Settings***

| NIB attribute | Identifier | Initial Value |
|---|---|---|
| nwkDiscoveryLQIThreshold | 0x62 | 0 (no filtering at NWK layer) |
| nwkDiscoveryRepetitionInterval | 0x63 | 0x00927c (600 ms)* |
| nwkIndicateDiscoveryRequest | 0x66 | TRUE |
| nwkMaxDiscoveryRepetitions | 0x69 | 0x02 |
| nwkMaxReportedNodeDescriptors | 0x6c | 16 |
| *Some timing values within this specification are specified in terms of MAC symbols. One MAC symbol is equal to 16µs. Where appropriate, absolute time values are presented in both MAC symbols and actual time in parentheses. | | |

All nodes operating according to the RF4CE MSO Profile SHALL support security.

All nodes SHALL ensure that the security capable sub-field of *nwkcNodeCapabilitie*s is set to one (1).

### 6.4.2 Binding Procedure

This procedure specifies how controller and target devices set up a link.

The pairing procedure SHALL be initiated by the controller device.

Target devices SHALL NOT bind with each other.

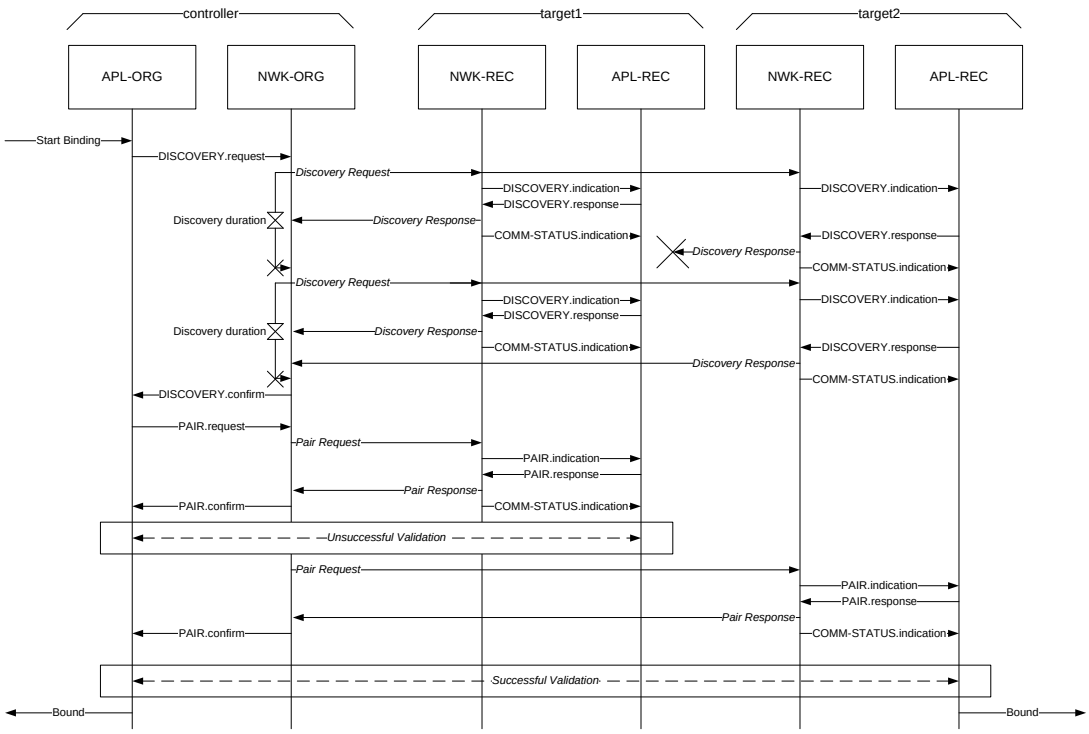


***Figure 1 - RF4CE MSO Profile Binding Protocol***

As shown in Figure 1, the binding procedure SHALL comprise three procedures:

- Discovery procedure: the controller will search and rank potential targets to pair with.
- Temporary pairing procedure: the controller will setup a link with the target that has the highest ranking determined in the discovery procedure.
- Validation procedure: the temporary link is used to check whether the ranking resulted in the desired selection. If not, the procedure will attempt the temporary pairing procedure with the next highest-ranked target.

The controller SHALL have three (3) different binding states on the controller side that correspond to the states specified in Table 15 and adhere to the state diagram in Figure 2.

***Table 15 - Binding States of a Controller at the Controller Side***

| Binding State | Description | Behavior |
|---|---|---|
| Not bound | The controller did not attempt to bind with a target yet.<br>OR<br>The controller did not succeed to bind with a target, and the controller was not bound to the same or other target before.<br>OR<br>The controller unbound from the target. | The NWK layer pairing table is empty. As such, no data frames can be sent to any target. |
| Binding | The controller is busy with the binding procedure: The controller has temporarily paired with the target, but has not yet been informed that the validation procedure completed successfully or unsuccessfully. | Once the target is in the NWK layer pairing table, the controller can send data frames to the target that will only be used for validation purposes. |

| Binding State | Description | Behavior |
|---|---|---|
| Bound | The controller successfully bound with the target: The controller has temporarily paired with the target, and has been informed that the validation procedure completed successfully.<br>OR<br>The controller did not succeed to bind with the target, but it was bound to the same or other target before, and this state was restored. | The target is in the NWK layer pairing table and the controller can send data frames to the target that can be used for all purposes. |

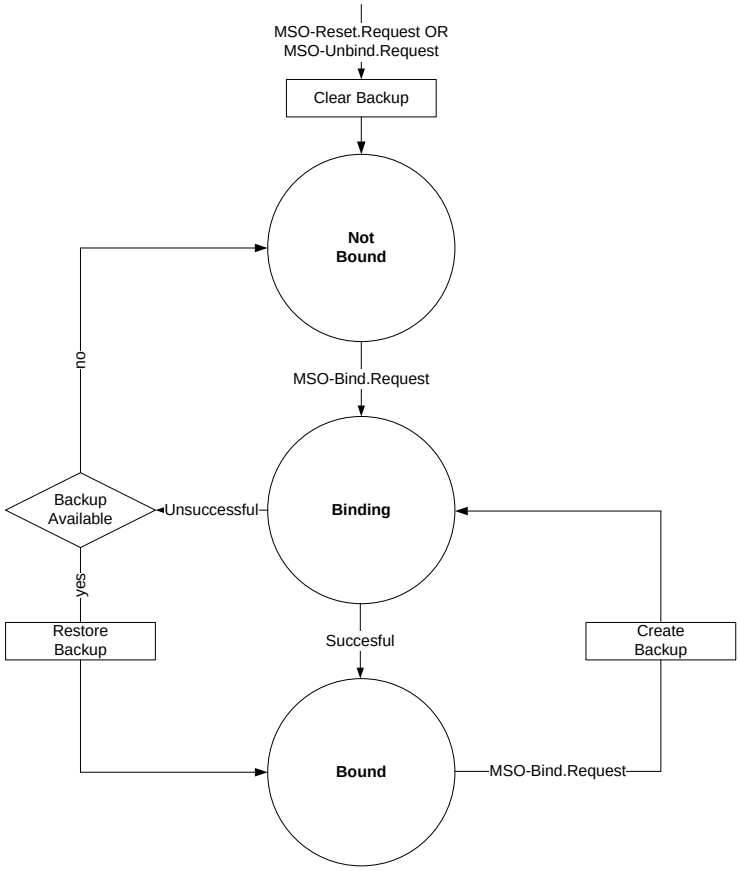


***Figure 2 - Controller's Binding State Diagram at the Controller Side***

The controller SHALL have six (6) different validation states on the target side that correspond to the states specified in Table 16 and adhere to the state diagram in Figure 2.

***Table 16 - Validation States of a Controller at the Target Side***

| Validation State | Description | Behavior |
|---|---|---|
| Not Validated | The controller has not (temporarily) paired with the target, and as such is not part of the target's NWK layer pairing table.<br>OR<br>The controller unbound from the target using the NLME-UNPAIR.request primitive. An RF4CE-NLME-UNPAIR.Indication has been triggered at the target side and the controller's pairing entry in the target's NWK layer pairing table has been deleted accordingly.<br>OR<br>The target unbound from the controller. The controller's pairing entry in the target's NWK layer pairing table has been deleted accordingly.<br>OR<br>The controller has temporarily paired with the target, but completed the validation procedure unsuccessfully. This has been communicated to the controller, and its pairing entry in the target's NWK layer pairing table has been deleted accordingly. | The controller is not in the NWK layer pairing table. As such all data frames that the target receives from the controller will be dropped. |
| Validating | The controller has temporarily paired with the target, but has not yet completed the validation procedure. In addition, the controller was not validated with the target before this operation. | The controller is in the target's NWK layer pairing table. All data frames that the target receives from the controller will only be used for validation purposes. |
| Rejected | The controller has temporarily paired with the target, but completed the validation procedure unsuccessfully. This has not yet been communicated to the controller however. | The controller is in the target's NWK layer pairing table. All data frames that the target receives from the controller will be ignored, except the check validation request frames which trigger the check validation response frames that communicate the unsuccessful validation result back to the controller. |
| Validated | The controller has temporarily paired with the target, and has successfully completed the validation procedure. | The controller is in the target's NWK layer pairing table. All data frames that the target receives from the controller can be used for all purposes. |
| Revalidating | The controller has temporarily paired with the target, but has not yet completed the validation procedure. The controller was validated with the target before this operation. | The controller is in the target's NWK layer pairing table. All data frames that the target receives from the controller will only be used for validation purposes. |
| Previously Validated | The controller has temporarily paired with the target, but completed the validation procedure unsuccessfully. The controller was validated with the target before this operation. | The controller is in the target's NWK layer pairing table. All data frames that the target receives from the controller can be used for all purposes. |

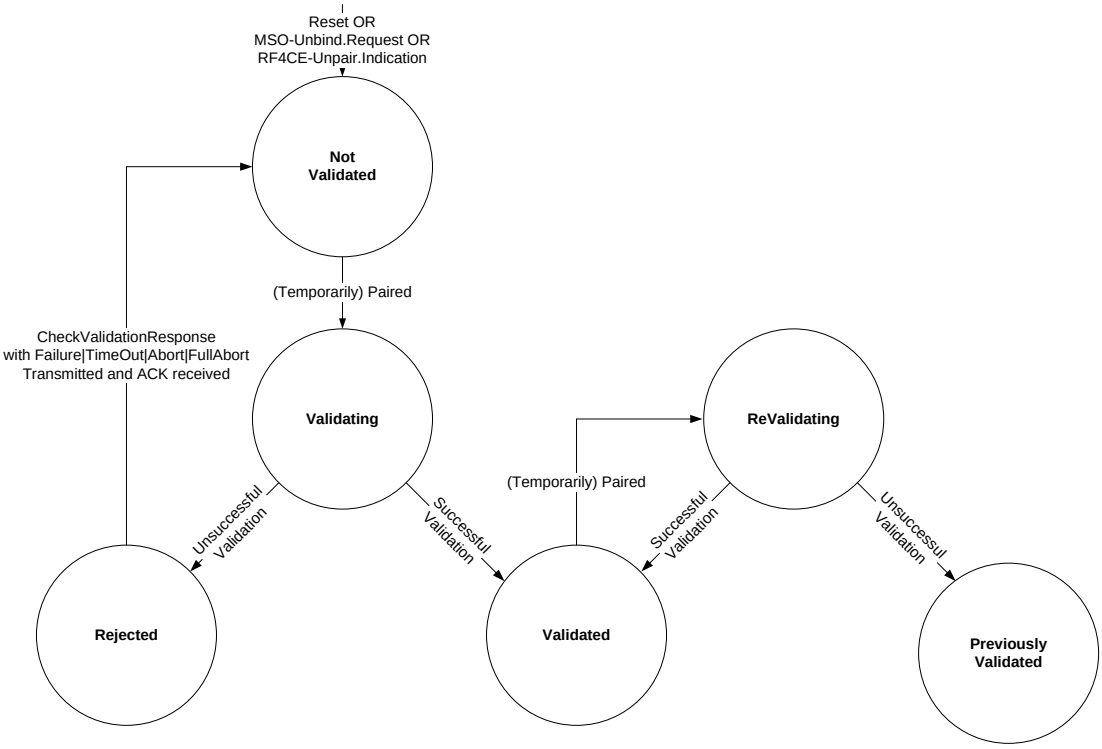


***Figure 3 - Controller's Validation State Diagram at the Target Side***

## 6.5 Controller Side Binding Procedure

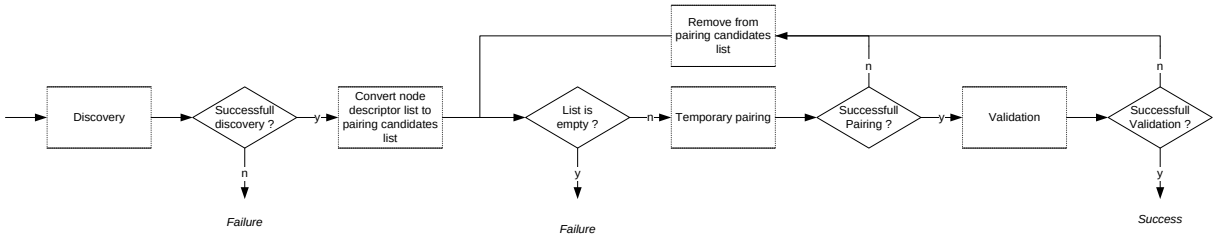


***Figure 4 - Binding Procedure Controller Flowchart***

Figure 4 shows the binding procedure at the controller side.

The controller SHALL implement the procedure specified in Figure 4.

When the binding procedure is initiated, it SHALL trigger the network layer by issuing the NLME-DISCOVERY.request primitive to the NLME.

For the discovery, the values *of nwkDiscoveryRepetitionInterval*, *nwkMaxDiscoveryRepetitions* and *nwkMaxReportedNodeDescriptors* SHALL be defined as specified in Table 13.

The DiscDuration parameter of the NLME-DISCOVERY.request primitive SHALL be set to 0x0064 (100 ms).

The profile identifier list disclosed as supported by the node SHALL contain only the value 0xc0, the RF4CE MSO Profile identifier.

The list of profile identifiers by which incoming discovery response command frames are matched SHALL contain only the value 0xc0, the RF4CE MSO Profile identifier specified in [RF4CE ID].

The Requested device type SHALL be specified according to [RF4CE TYPE].

On receipt of an unsuccessful discovery confirmation from the network layer through the NLME-DISCOVERY.confirm primitive, the profile SHALL terminate the procedure.

On receipt of a successful discovery confirmation from the network layer through the NLME-DISCOVERY.confirm primitive, containing one or more node descriptors, the profile SHALL rank, filter and truncate the list of received node descriptors to form the pairing candidates list according to the procedure specified in Section 6.5.4.

The profile of the controller SHALL trigger a pair request by issuing the NLME-PAIR.request primitive to the NLME to set up a temporary pairing with the target with the highest ranking in the pairing candidates list.

On receipt of an unsuccessful confirmation of the pairing, the profile of the controller SHALL attempt to setup a new temporary pairing with the next highest ranked target in the pairing candidates list.

If there are no more targets left in the pairing candidates list, the profile of the controller SHALL terminate the procedure.

On receipt of a successful confirmation of pairing from the network layer, via the NLME-PAIR.confirm primitive, the profile SHALL start the validation procedure specified in Section 6.5.6.

If the validation is unsuccessful ('Failure', 'Time Out', 'Collision', 'Abort', 'Full Abort'), the profile of the controller SHALL remove the pairing entry of the target from its pairing table.

The profile of the controller SHALL attempt to setup a new temporary pairing with the next highest ranked target in the pairing candidates list.

If there are no more targets left in the pairing candidates list, the profile of the controller SHALL terminate the procedure.

If the validation is unsuccessful with "Full Abort” status, the profile SHALL terminate the procedure.

If the validation is unsuccessful ('Failure', 'Time Out', 'Collision', 'Abort', 'Full Abort'), and the controller was bound with a target before the binding procedure started, the old binding SHALL be restored.

If the validation is successful, the pairing SHALL be considered permanent and the controller is bound with the target.

The controller SHALL set the key exchange transfer count to *aplKeyExchangeTransferCount* for each temporary pairing procedure.

### 6.5.1 Target Side Binding Procedure

During initialization of a target device, the NIB attribute *nwkIndicateDiscoveryRequests* SHALL be set to TRUE.

On receipt of a discovery indication notification from the NLME through the NLME-DISCOVERY.indication primitive, the profile of the target SHALL reply with a successful discovery response if all the received discovery indications meet the following conditions:

- The Vendor identifier matches the Vendor ID of the target device (see list of Vendor IDs in [RF4CE CODE]
- The RF4CE MSO Profile (0xc0) is specified in [RF4CE ID].
- The Requested Device Type matches device type of target device or is set to 0xFF.

If one of these conditions is not met, the target profile SHALL ignore the discovery indication.

After a transmission of a successful discovery response to the NLME, the target profile SHALL wait for a corresponding pairing request notification from the NLME through the NLME-PAIR.indication primitive.

The profile SHALL respond to the pair request, and SHALL notify the network layer by issuing the NLME-PAIR.response primitive to the NLME.

If the transmission of the subsequent pair response command and subsequent link key exchange – if required – was successful, indicated through the NLME-COMM-STATUS.indication primitive from the NLME, the profile SHALL be considered temporary paired to the device with the indicated reference into the pairing table.

After a successful temporary pairing, the target profile SHALL enter the validation procedure specified in Section 6.5.6.

If the validation is unsuccessful ('Failure', 'Time Out', 'Collision', 'Abort', 'Full Abort'), the target SHALL remove the pairing entry of the controller from its pairing table only and only if the controller was not validated before the binding procedure started.

If the validation is 'Successful', the binding SHALL be considered permanent and the target is bound with the controller.

If the target receives a new discovery or pair request with the conditions specified in requirement MP-50-220 during the validation procedure, the validation SHALL be terminated.

The target SHALL remove the pairing entry of the controller from its pairing table only and only if the controller was not validated before the binding procedure started.

A garbage collection algorithm SHALL be used to avoid the condition of the pairing table of the target being filled with old controllers, preventing new controllers from binding.

The exact implementation of the garbage collection algorithm SHALL be implementation-specific.

### 6.5.2 Controller Side Status Sharing in Discovery Requests

To allow the target to adapt its discovery responses to the status of the controller, the controller nodes SHALL add extra status information to the user string field of the discovery request.

The 15 octets wide user string SHALL be formatted as illustrated in Table 17.

The first nine (9) octets SHALL contain the MSO user string, which is used to carry application related information and is ignored completely by the prioritization process.

The MSO user string SHALL be ASCII-encoded.

The tenth octet SHALL be a Null byte, which formally indicates the end of the MSO user string, ensuring that non-MSO-profile-based platforms will not try to interpret the next five octets, which will not be ASCII encoded.

The next four octets SHALL be reserved for future use.

The final octet SHALL contain the Binding Initiation Indicator, which indicates what started the binding process.

***Table 17 - Format of the Discovery Request User String***

| Octet 0-8 | Octet 9 | Octet 10-13 | Octet 14 |
|---|---|---|---|
| MSO User String | Null | Reserved | Binding Initiation Indicator |

#### *6.5.2.1 MSO User String*

The MSO user string field SHALL be nine (9) octets in length.

The MSO user string field SHALL be empty by default if no user string value is provided by the controller.

#### *6.5.2.2 Binding Initiation Indicator*

The Binding Initiation Indicator SHALL be one (1) octet in length.

The Binding Initiation Indicator SHALL specify how the binding process was initiated on the controller.

Binding Initiation information MAY be used by the target when generating the class descriptors in the user string of the discovery response.

Possible values for the Binding Initiation Indicator SHALL be as specified in Table 18.

***Table 18 - Binding Initiation Indicator***

| Value | Scenario | Description |
|---|---|---|
| 0x00 | Dedicated Key Combo Bind | Binding process is initiated by a dedicated key combination. |
| 0x01 | Any Button Bind | Binding process is initiated by any button press, while the controller is in an unpaired state. |
| 0x02-0xFF | Reserved | Reserved for future use. |

### 6.5.3 Target Side Status Sharing in Discovery Responses

To allow the controller to prioritize the node descriptors, the target nodes SHALL add extra status information to the User String field of the discovery response.

The User String SHALL be 15 octets in length.

The first nine (9) octets SHALL contain the MSO User String.

The MSO User String SHALL be ASCII-encoded.

The MSO User String MAY be used to carry application-related information.

The tenth octet SHALL be a Null byte, which formally indicates the end of the MSO User String, ensuring that non-MSO-profile-based platforms will not try to interpret the next octets, which will not be ASCII encoded.

The next three (3) octets of the User String SHALL contain class descriptors, which are used to rank the node descriptors.

The final two (2) octets SHALL contain Discovery LQI Thresholds.

Discovery LQI Thresholds MAY be used to remove non-compliant node descriptors from the list.

The User String SHALL be formatted as illustrated in Table 19.

***Table 19 - Format of Discovery Response User String***

| Octet 0-8 | Octet 9 | Octet 10 | Octet 11 | Octet 12 | Octet 13 | Octet 14 |
|---|---|---|---|---|---|---|
| MSO User String | Null | Tertiary Class Descriptor | Secondary Class Descriptor | Primary Class Descriptor | Strict LQI Threshold | Basic LQI Threshold |

#### *6.5.3.1 MSO User String*

The MSO User String SHALL be nine (9) octets in length.

Characters within each octet of the MSO User String SHALL be ASCII-encoded.

The MSO User String MAY be used by the application to carry application related information.

#### *6.5.3.2 Basic Discovery LQI Threshold*

The Basic Discovery LQI Threshold SHALL be one (1) octet in length.

If the LQI value of the discovery response for a node descriptor is less than the Basic Discovery LQI Threshold, that node descriptor SHALL be removed from the list as specified in Section 6.5.4.

#### *6.5.3.3 Strict LQI Threshold*

The Strict LQI Threshold SHALL be one (1) octet in length.

A node descriptor SHALL be removed from the list based on the Strict LQI Threshold if both of the following are true:

The LQI value of the discovery response for a node descriptor is less than the Strict Discovery LQI Threshold.

Adherence to the Strict LQI Threshold SHALL be required according to the applicable class descriptor as specified in Section 6.5.4.

#### *6.5.3.4 Primary Class Descriptor*

The Primary Class Descriptor SHALL be one (1) octet in length.

The Primary Class Descriptor SHALL contain four (4) fields as shown in Table 20.

***Table 20 - Primary Class Descriptor***

| Bit 0-3 | Bit 4-5 | Bit 6 | Bit 7 |
|---|---|---|---|
| Class Number | Duplicate Class Number Handling | ApplyStrict LQIThreshold | EnableRequest AutoValidation |
| | | | |

##### *6.5.3.4.1 Class Number*

The Class Number field SHALL be four (4) bits in length.

The Class Number field SHALL be used to rank the node descriptors in the list.

A lower Class Number SHALL result in a better ranking as specified in Section 6.5.4.

##### *6.5.3.4.2 Duplicate Class Number Handling*

The Duplicate Class Number Handling field SHALL be two (2) bits in length.

The Duplicate Class Number Handling field SHALL indicate what should happen with the node descriptor if another node descriptor with the exact same class number is discovered by the controller.

Possible Duplicate Class Number Handling values SHALL include those specified in Table 21.

***Table 21 - Duplicate Class Number Handling***

| Bit 5:4 | Description |
|---|---|
| 00 | Use node descriptor as is |
| 01 | Remove node descriptor |
| 10 | Reclassify node descriptor |
| 11 | Abort binding |

##### *6.5.3.4.3 Apply Strict LQI Threshold*

The Apply Strict LQI Threshold field SHALL be one (1) bit in length.

The Apply Strict LQI Threshold field SHALL indicate if the Strict LQI Threshold should be applied to a node descriptor.

If this bit is set and the LQI of the discovery response for a node descriptor is lower than the Strict LQI Threshold, the node descriptor SHALL be removed from the list as specified in Section 6.5.4.

##### *6.5.3.4.4 Enable Request Auto Validation*

The Enable Request Auto Validation field SHALL be one (1) bit in length.

The Enable Request Auto Validation field SHALL indicate if the check validation request that the controller will send to the target during the validation procedure will have the Request Auto Validation bit set, as specified in Section 6.5.4.

#### 6.5.3.5 Secondary Class Descriptor

The Secondary Class Descriptor SHALL be formatted identically to the Primary Class Descriptor.

The Secondary Class Descriptor SHALL only be used when both of the following are true:

- The Node Descriptor needs to be reclassified because the class number of the Primary Class Descriptor is not unique.
- The 'Duplicate class number handling' field of the Primary Class Descriptor indicates 'Reclassify node descriptor'.

#### 6.5.3.6 Tertiary Class Descriptor

The Tertiary Class Descriptor SHALL be formatted identically to the primary (and secondary) class descriptors.

The Tertiary Class Descriptor SHALL only be used when both of the following are true:

- The Node Descriptor needs to be reclassified because the class numbers of both the Primary and Secondary Class Descriptors are not unique.
- The 'Duplicate class number handling' field of both the Primary and Secondary Class Descriptors indicate 'Reclassify node descriptor'.

The 'Duplicate class number handling' field of the Tertiary Class Descriptor SHOULD NOT indicate 'Reclassify node descriptor'.

If the 'Duplicate class number handling' field of the Tertiary Class Descriptor indicates 'Reclassify node descriptor', the target SHALL interpret it as 'Remove node descriptor'.

### 6.5.4 From Node Descriptor List to Pairing Candidates List

The discovery procedure on a controller SHALL yield a list of node descriptors.

This list of node descriptors SHALL be ranked, filtered and truncated to yield the pairing candidates list.

#### 6.5.4.1 Basic LQI Threshold Filtering

The controller SHALL remove all node descriptors for which the LQI of the discovery response is below the 'Basic LQI Threshold' as indicated in the user string of the discovery response.

#### 6.5.4.2 Initial Ranking

The controller SHALL rank the node descriptors based on the class number in the Primary Class Descriptor of the User String of the Discovery Response.

There SHALL be an inverse relationship between a node descriptor's class number and its position in the node descriptor list.

If two controllers have the same class number, the LQI value SHALL be used as the tiebreaker.

#### 6.5.4.3 Duplicate Class Number Handling

The controller SHALL check the list for node descriptors with non-unique class numbers

Node descriptors with non-unique class numbers SHALL be handled according to the 'Duplicate Class Number Handling' field in the Primary Class Descriptor.

Handling of node descriptors with non-unique class numbers in the Primary Class Descriptor SHALL result in one of the following solutions:

- Binding procedure is aborted
- Node descriptor is kept in list
- Node descriptor is removed from list
- Node descriptor is reclassified

If a node descriptor is reclassified from the Primary Class Descriptor, it SHALL be re-ranked based upon the class number in the Secondary Class Descriptor.

If reclassification from the Primary Class Descriptor results in non-unique class numbers in the Secondary Class Descriptor, the 'Duplicate Class Number Handling' field of the Secondary Class Descriptor SHALL be used to handle them.

Handling of node descriptors with non-unique class numbers in the Secondary Class Descriptor SHALL result in one of the solutions specified above for handling of non-unique class numbers in the Primary Class Descriptor.

If a node descriptor is reclassified from the Secondary Class Descriptor, it SHALL be re-ranked based on the class number in the Tertiary Class Descriptor.

If reclassification from the Secondary Class Descriptor results in non-unique class numbers in the Tertiary Class Descriptor, the 'Duplicate Class Number Handling' field of the Tertiary Class Descriptor SHALL be used to handle them.

Handling of node descriptors with non-unique class numbers in the Tertiary Class Descriptor SHALL result in one of the following solutions:

- Binding procedure is aborted
- Node descriptor is kept in list
- Node descriptor is removed from list

#### *6.5.4.4 Strict LQI Threshold filtering*

If 'Apply Strict LQI Threshold' is set in the final class descriptor that was used in the ranking process, the controller SHALL remove all node descriptors for which the LQI of the discovery response is below the 'Strict LQI threshold' after duplicate class number handling.

If 'Enable Request Automatic Validation' is set in the final class descriptor that was used in the ranking process for a node descriptor, the check validation request frames in the validation procedure for this node descriptor SHALL have the 'Request Automatic Validation' flag set.

#### *6.5.4.5 Truncating*

After the received node descriptors are ranked and filtered, the list SHALL be truncated.

Only the top aplMaxPairingCandidates node descriptors SHALL form the pairing candidates list.

### 6.5.5 Class Descriptor Generation at Target Side

When a target receives a Discovery Request from a controller, the target SHALL generate the Discovery Response.

The Discovery Response generated by the target SHALL include the user string that contains up to three class descriptors.

Each of the class descriptors SHALL represent a certain scenario.

The generation of the class descriptors SHALL be implementation-specific.

### 6.5.6 Detailed Validation Procedure

The validation procedure SHALL provide a mechanism to check if a temporary pairing of a controller and a target can be considered as permanent or not permanent.

How the validation is handled by the target is out of the scope of this specification. As shown in Figure 5 through Figure 9, this specification only defines the hooks that the application can use to make a validation procedure that suits its requirements with respect to user friendliness, availability of buttons, availability of user interaction, etc. Note that any differences between variants are only present at the target side, and are completely transparent for the controller.

The validation procedure starts with an *aplcBlackOutTime* interval during which both controller and target SHALL NOT transmit any packets, allowing the implementation to complete the temporary pairing.

After the *aplcBlackOutTime* interval, the validation procedure MAY start with an optional "Validation Configuration RIB retrieval" step which exchanges updated values of the properties used for the validation procedure on the controller side.

These updated properties, if used, SHALL be stored on the target and can be retrieved by the controller via a RIB request as specified in Section 6.5.8.2.

If this property exchange is omitted or unsuccessful, the controller SHALL continue the validation procedure with its default parameters.

To verify if the validation is successful, the controller SHALL generate and transmit a Check Validation Request command frame to the target device every *aplAutoCheckValidationPeriod*.

- If Validation Configuration RIB Retrieval is not attempted, the first Check Validation Request SHALL be sent *aplAutoCheckValidationPeriod* after the blackout interval.
- If Validation Configuration RIB Retrieval is attempted, the first Check Validation Request SHALL be sent *aplAutoCheckValidationPeriod* after the completion of the Validation Configuration RIB Retrieval.

If the transmission of the check validation request command frame was successful, the controller SHALL wait *aplcResponseIdleTime* with its receiver disabled.

After waiting *aplcResponseIdleTime*, the controller SHALL enable its receiver and wait *aplResponseWaitTime* symbols for the corresponding check validation response command frame to arrive.

Upon receipt of a check validation request command frame, the target SHALL check the result of the validation.

After the *aplcResponseIdleTime* interval, the target SHALL generate and transmit a check validation response command frame to the controller device, containing the result of the validation, as specified in Table 7.

If the controller receives the Check Validation Response command frame, it SHALL disable the receiver.

If the ValidationStatus field was set to 'Success', the validation SHALL be successful.

If the ValidationStatus field is set to 'Failure', 'Time Out', Collision', 'Abort' or 'Full Abort', the validation SHALL fail.

If the ValidationStatus field is set to Pending, the validation SHALL continue.

If the controller doesn't receive the check validation response command frame within the *aplResponseWaitTime* interval, the receiver SHALL be disabled.

If the first validation watchdog kick is not performed on the target in the a*plInitialValidationWatchdogTime* interval (starting at the beginning of the validation procedure), the validation SHALL fail (Time Out).

After each validation watchdog kick, the watchdog time out interval SHALL be restarted with the value specified in the validation watchdog kick.

If the validation step is not performed on the target in the *aplValidationWaitTime* interval (starting at the beginning of the validation procedure), the validation SHALL fail (Time Out).

A target MAY also terminate the validation procedure on its own behalf (for example if a user control frame is received from a previously bound controller).

If the target terminates the validation phase, the validation SHALL fail (Failure).

If a (Full) AbortValidation key is supported, the validation MAY be fully aborted by the controller.

When the (Full) AbortValidation key is pressed, the controller SHALL transmit the key to the target via a user control frame.

The target SHALL interpret the key as a (full) abort request, and the validation SHALL be unsuccessful.

This SHALL be communicated back to the controller via the check validation request/responses.

The ValidationStatus field MAY be set to either:

- 'Abort', to abort the validation with this target only. In this case, the validation is unsuccessful.
- 'Full Abort', to abort the complete binding procedure. In this case, the validation is unsuccessful.

The controller SHALL also terminate the validation automatically if it detects that it has lost its link with the target for *aplLinkLostWaitTime*.

The *aplLinkLostWaitTime* timeout SHOULD be started if it is not already pending and when a command frame is transmitted to the target.

If a transmission of the command frame is successfully confirmed, the timeout SHALL be cancelled.

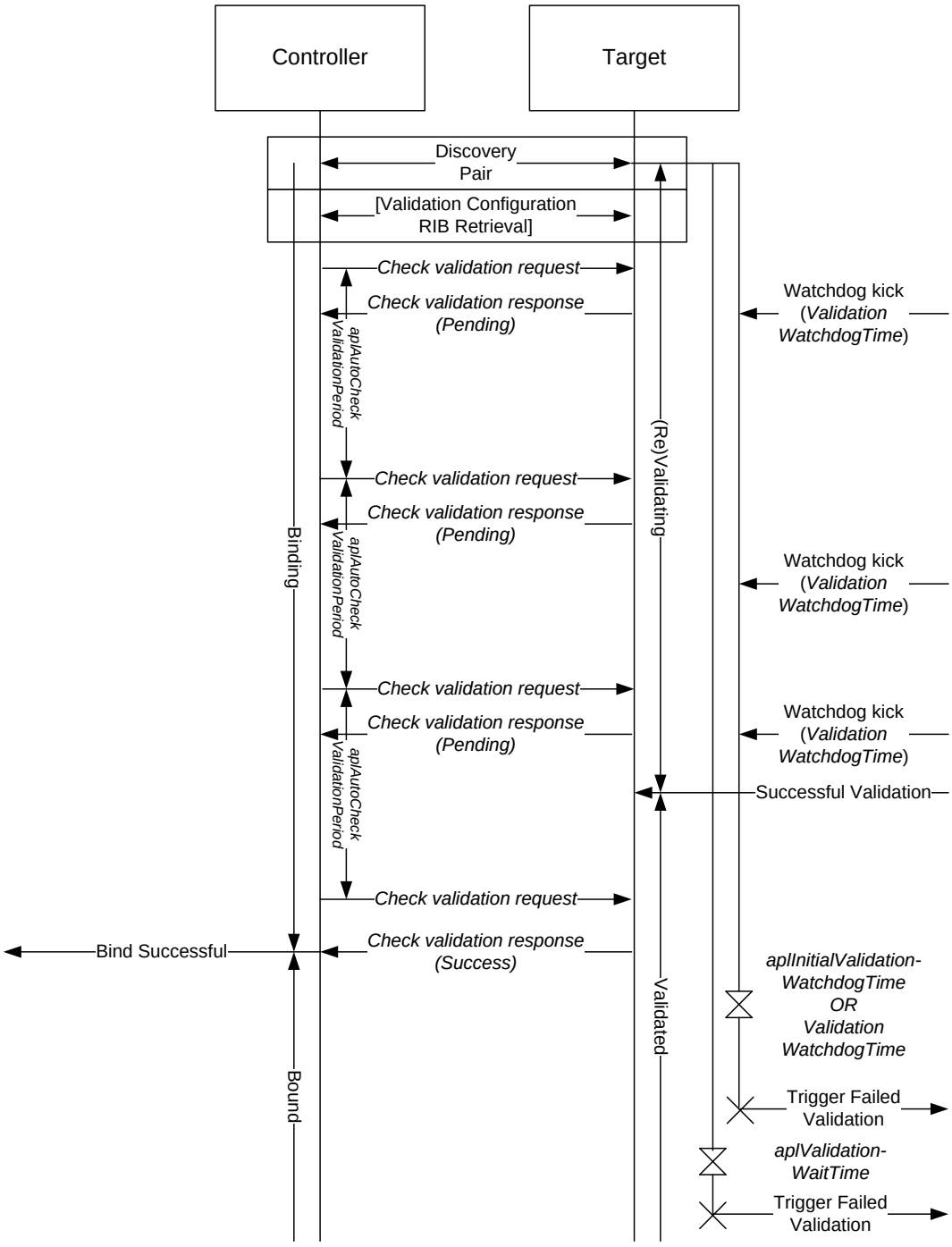


***Figure 5 - Validation Procedure Succeeded***

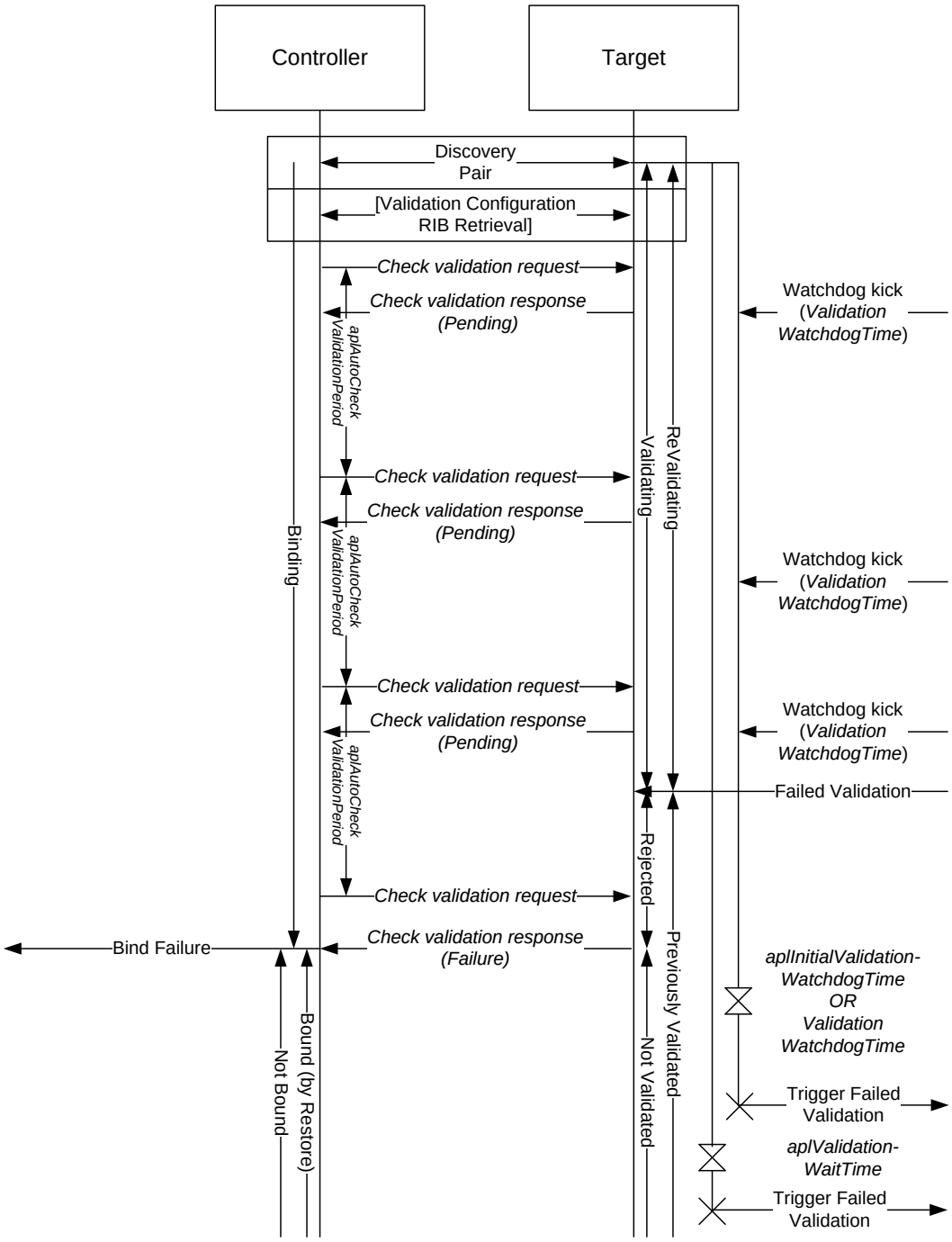


***Figure 6 - Validation Procedure Failed***

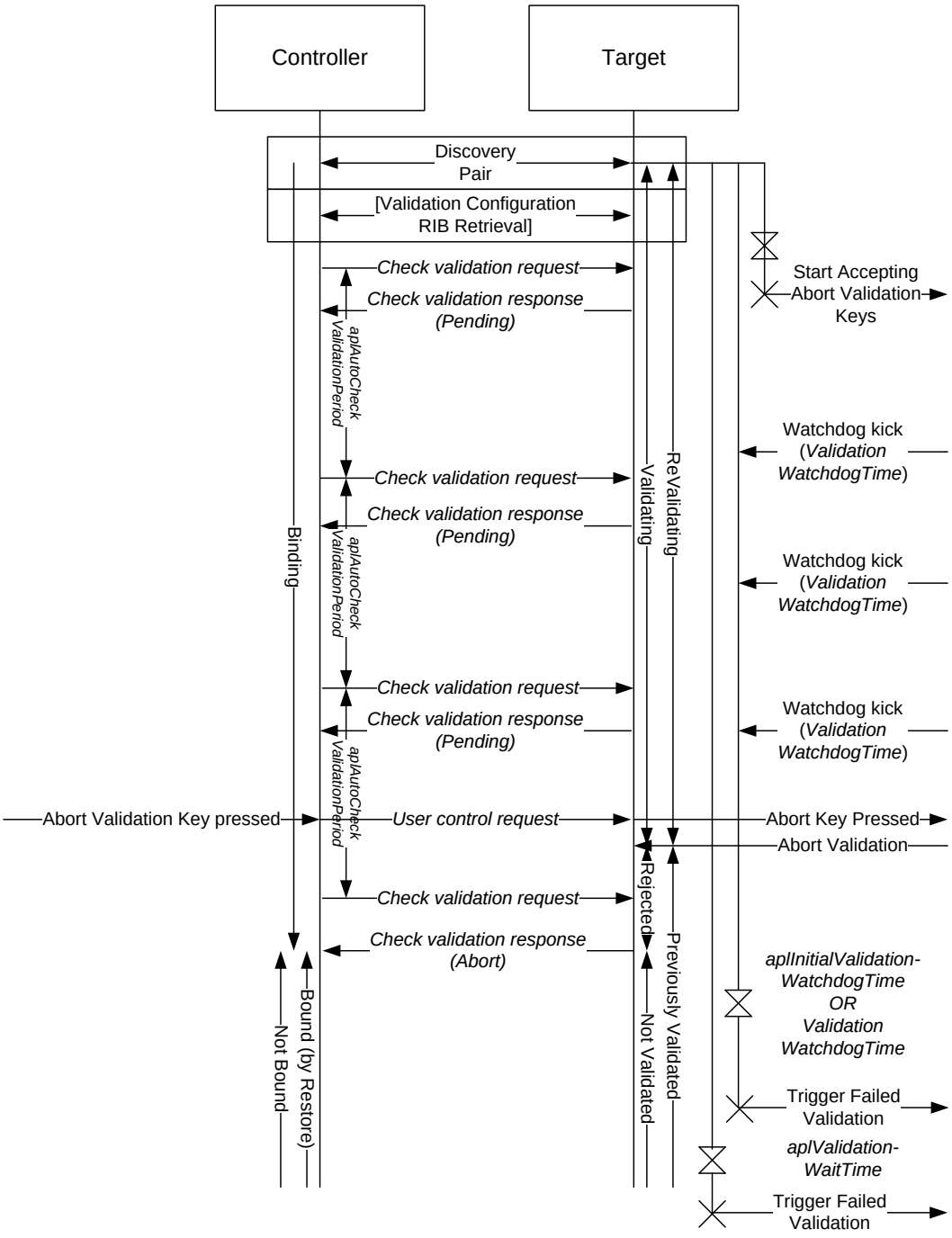


***Figure 7 - Validation Procedure Aborted by Controller***

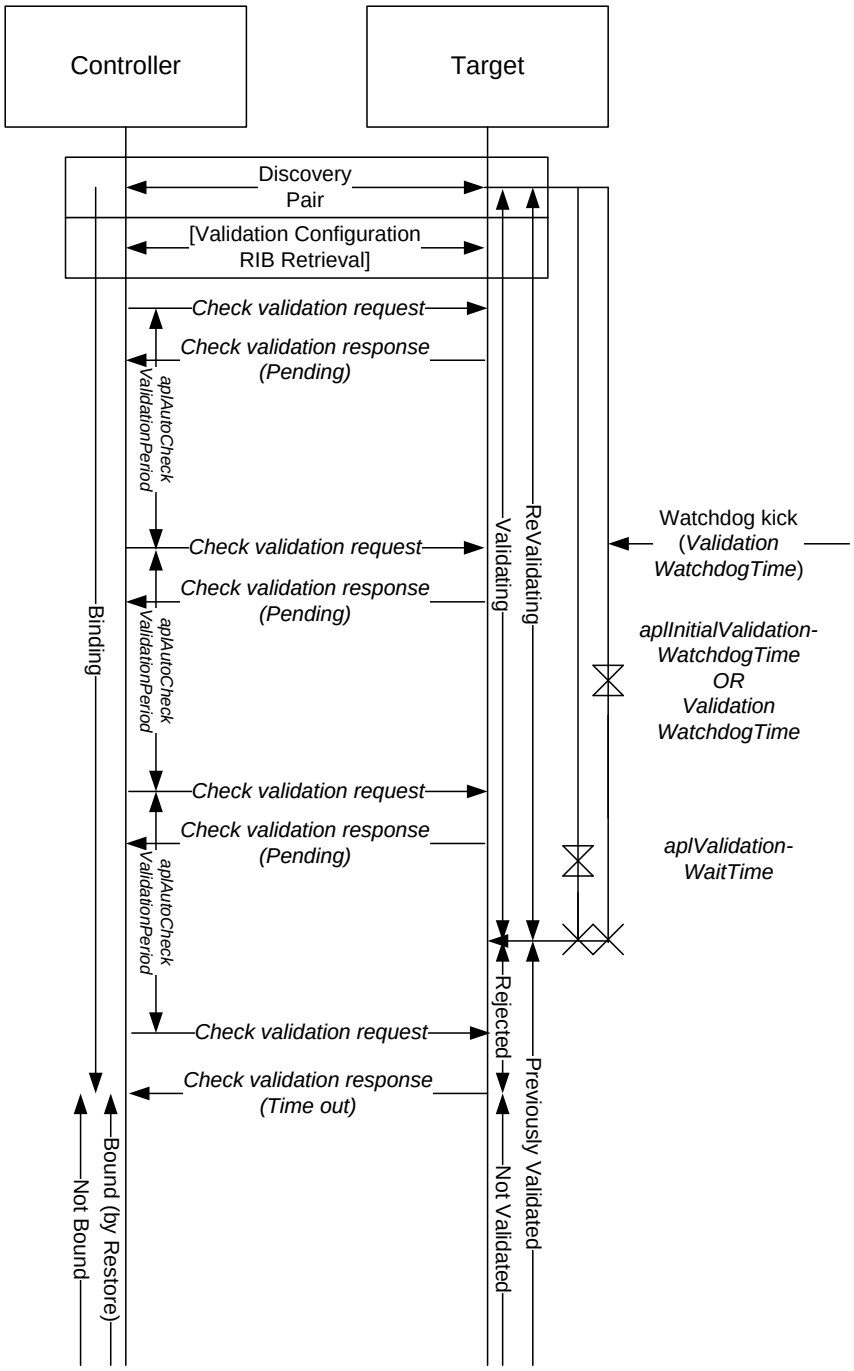


***Figure 8 - Validation Procedure Timed Out at Target Side***

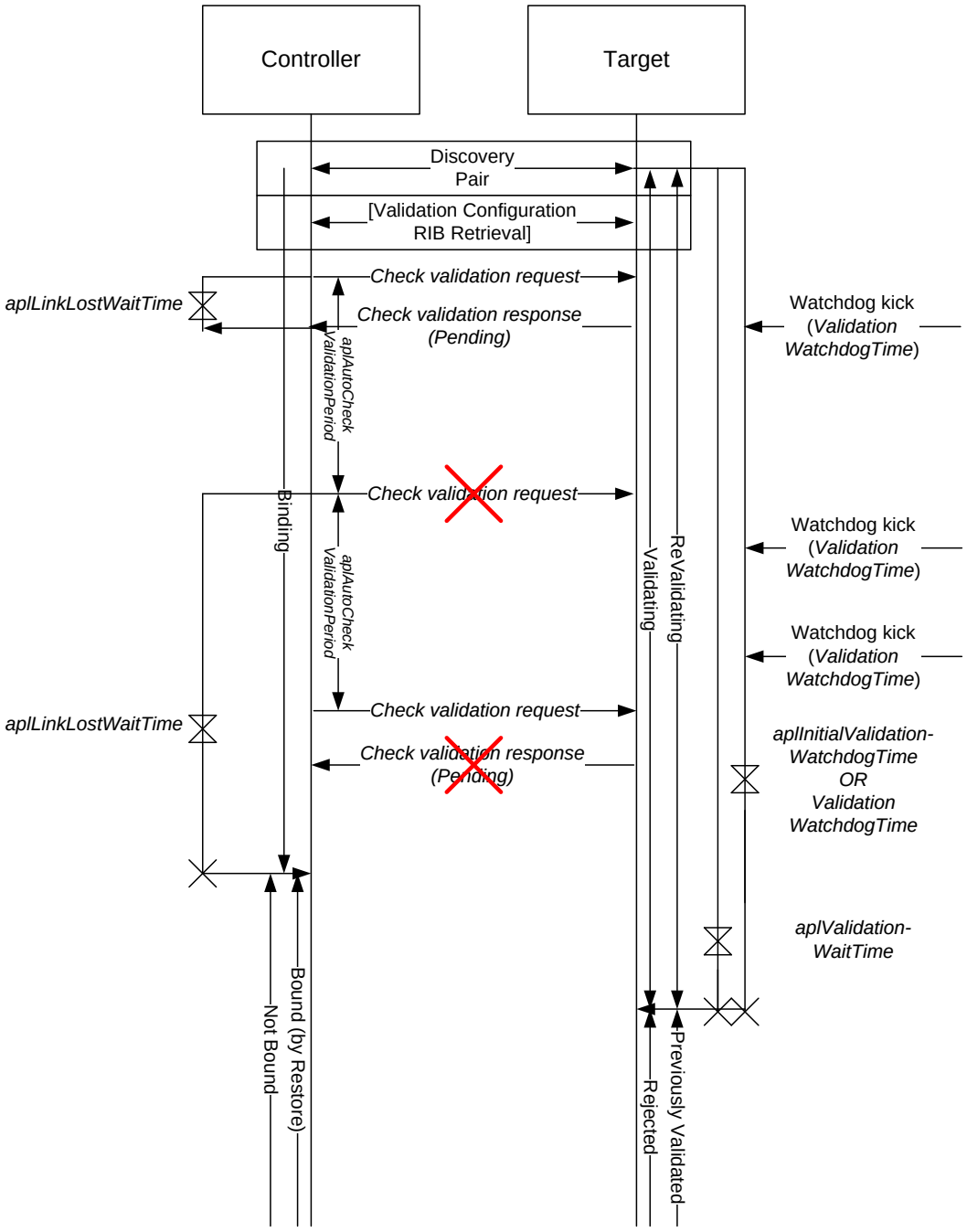


***Figure 9 - Validation Procedure Terminated by aplLinkLostWaitTime at Controller Side***

For illustrative purposes, two variants of an actual validation procedure are specified in this section as well: a buttonless variant and a button based variant.

#### *6.5.6.1 The Buttonless Variant*

In the buttonless variant of the validation procedure, the target application SHALL define the golden validation code as a random alphabetic or numeric value.

The UI on the target SHALL show this golden validation code.

The user is requested to enter this validation code on the controller, which SHALL be transferred to the controller via the user control frames.

The combination of all the digits the user enters on the controller SHALL be the user validation code.

The user SHALL receive immediate feedback of the current status of the user validation code via the UI on the target.

The buttonless variant SHALL configure the target side timeouts.

The result of the check validation response SHALL depend on the comparison of the golden validation code with the user validation code, as specified in Table 22.

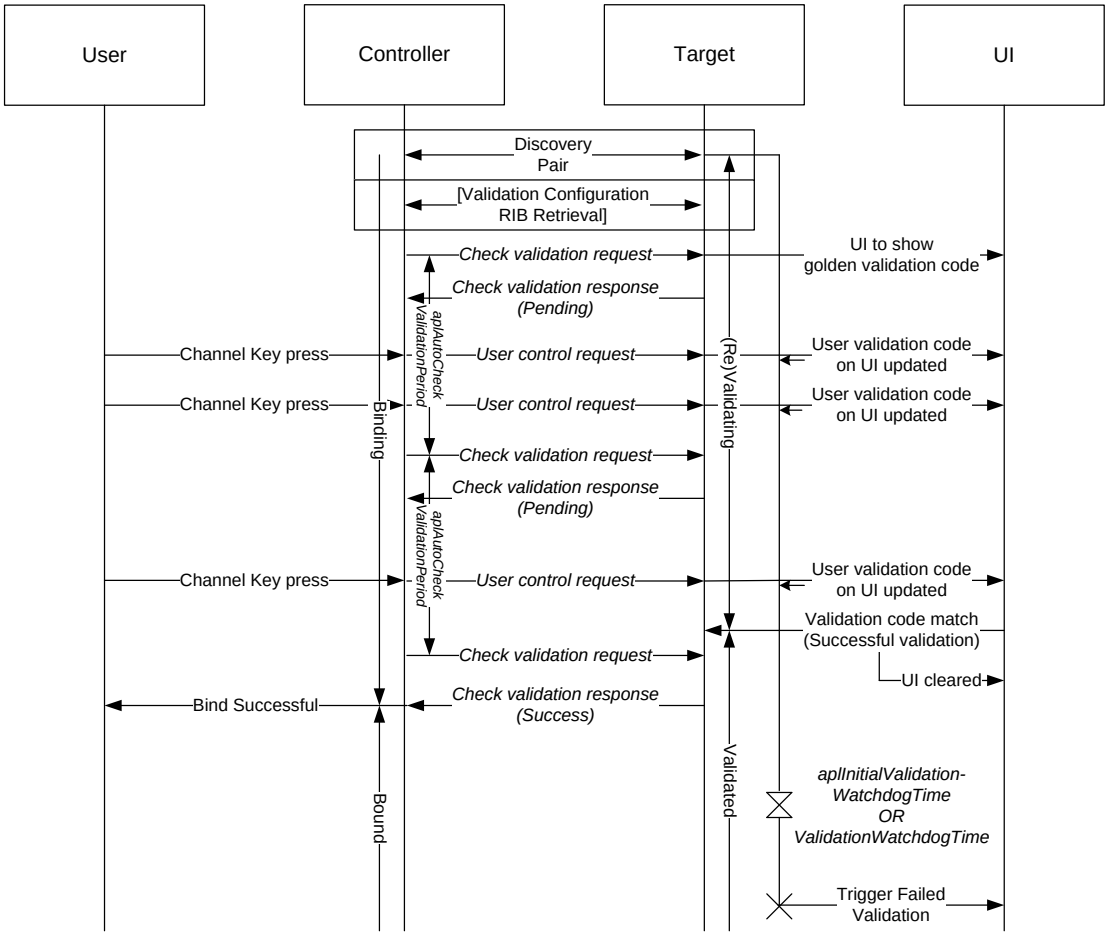


***Figure 10 - Buttonless Variant of Successful Validation Procedure***

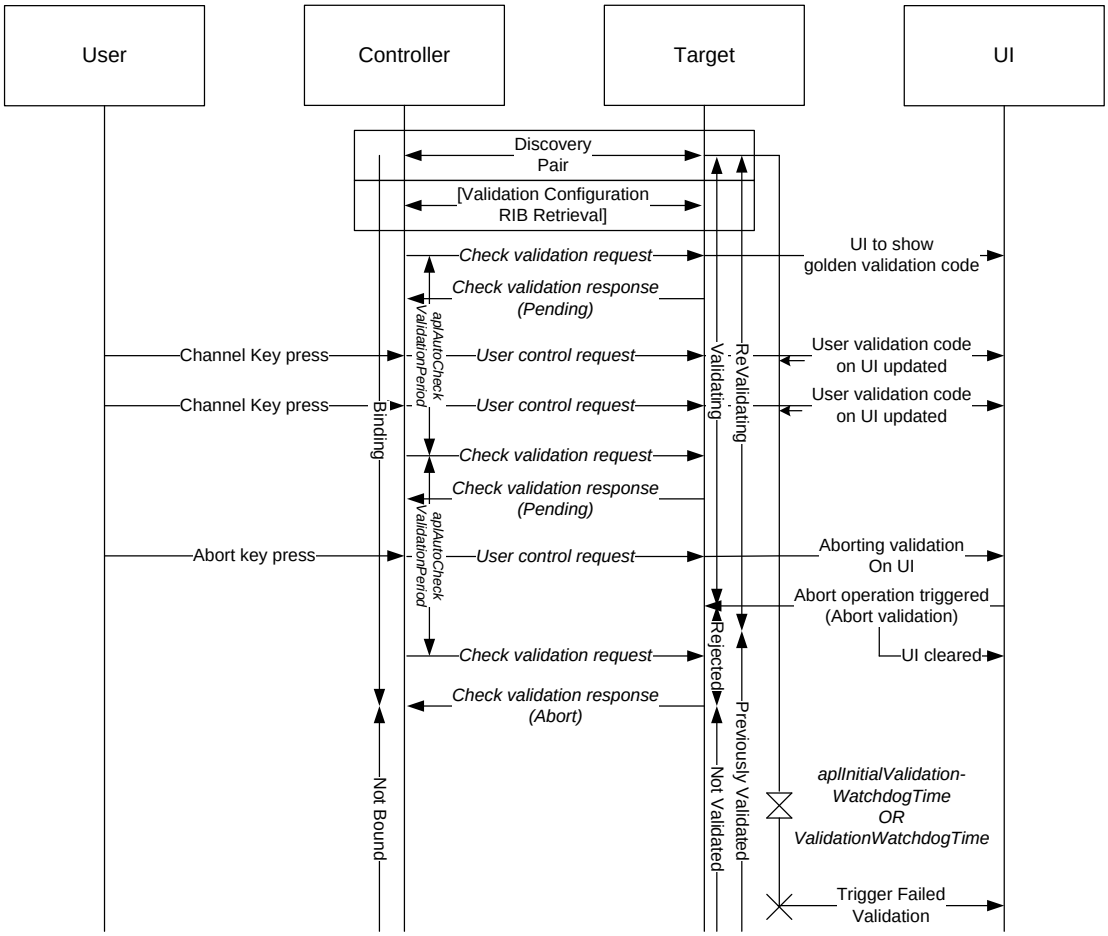


***Figure 11 - Buttonless Variant of Aborted Validation Procedure***

***Table 22 - Buttonless Validation Status Overview***

| Validation Status | Description |
|---|---|
| Success | The validation is successful. The reason for this is the following:<br>The user validation code was complete and the golden validation code and user validation code matched. |
| Failure | The validation failed. The reason for this is the following:<br>There was a mismatch between one of the entered user code digits and the golden validation code. |
| Time Out | The validation timed out. The reason for this is the following:<br>The watchdog triggered, since it was not kicked by the application, as the application did not receive a user control frame. The target has aborted the validation. |
| Collision | The validation was terminated because more than one controller simultaneously attempted to pair with the target. |
| Abort | The validation is aborted. The reason for this is the following:<br>The AbortValidation key is pressed on the controller. |
| Full Abort | The validation is fully aborted. The reason for this is the following:<br>The (Full)AbortValidation key is pressed on the controller. |

| Validation Status | Description |
|---|---|
| Pending | The validation is still in progress. The reason for this is the following:<br>The watchdog timer has not triggered yet, nor is the (Full)AbortValidation key pressed, nor is the user validation code complete. The user code digits that were entered so far do match with the leading digits of the golden validation code. |

### 6.5.7 User Control Procedure

The user control procedure of the RF4CE MSO Profile is very similar to the user control procedure of the ZRC profile specified in [RF4CE ZRC].

In the RF4CE MSO Profile the originator node of the user control commands SHALL be the controller and the recipient node SHALL be the target.

There is one important delta between the two profiles: unlike the ZRC profile not all keys are repeating. This results in a lower-energy application and is possible because the acknowledged transfers of RF4CE guarantee successful transmission of the key, if possible. The procedure for non-repeatable keys is specified in 6.5.7.2.

#### *6.5.7.1 User Control Procedure for Repeatable Keys*

The user control pressed command frame SHALL be supported as specified in [RF4CE ZRC].

#### *6.5.7.2 User Control Procedure for Non-Repeatable Keys*

##### *6.5.7.2.1 Originator Side User Control Procedure for Non-Repeatable Keys*

On receipt of a key press of a non-repeatable key from the user, the originator node SHALL generate and transmit a user control pressed command frame to the appropriate recipient node.

User control repeat command frames and user control released command frames SHALL NOT be sent.

A new user control pressed command frame for the same key SHALL NOT be sent until the key has been released and is pressed again.

##### *6.5.7.2.2 Recipient Side User Control Procedure for Non-Repeatable Keys*

On receipt of a user control pressed command frame for a non-repeatable key, the recipient node SHALL execute the requested operation once. Since no user control repeat or user control released command frames will be received for this key, no further processing is required.

#### *6.5.7.2.3 User Control Procedure for Non-Repeatable Keys Message Sequence Charts*

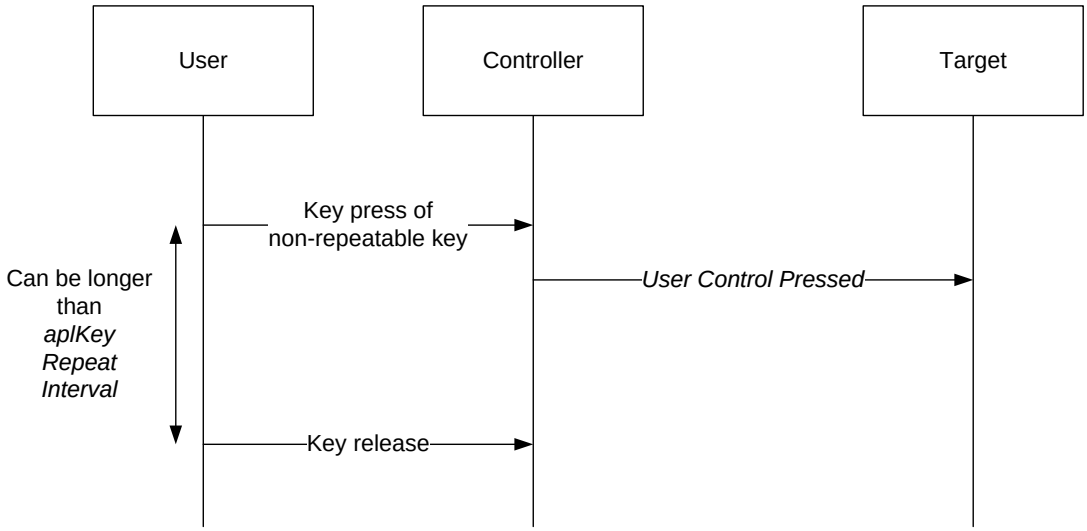


***Figure 12 - Basic User Control Procedure for Non-RepeatTable Key***

The basic control procedure after the press of a non-repeatable key SHALL be conducted as shown in Figure 12.

The user presses the non-repeatable key and releases this key after a period of time. Even when the period of time that the key is pressed is bigger than *aplKeyRepeatInterval*, a user control repeat command SHALL NOT be sent.

When the key is released, a user control released command SHALL NOT be sent.

### 6.5.8 Remote Storage Procedure

The RF4CE MSO Profile SHALL support controllers storing and retrieving data in the Remote Information Base (RIB) of the target.

A controller SHALL store data in the RIB through the set attribute request and response commands.

This data SHALL afterwards be retrievable from the RIB – by the same controller – using the get attribute request and response commands.

This data stored in the RIB SHALL also be accessible locally by the target itself.

Furthermore, controllers SHALL be able to retrieve data from the RIB that was stored there by the target using the same get attribute request and response commands.

Get/set attribute request/response frames over temporary paired links SHALL be ignored.

#### *6.5.8.1 Use Cases*

##### *6.5.8.1.1 Remote Storage of Peripheral IDs*

Identifiers for peripherals in the user's environment (e.g., brand and type of TV and/or other peripherals) SHALL be retrievable by a controller after battery replacement using the remote retrieve functionality.

Peripheral IDs MAY be stored there by the target or MAY be stored by the controller before the battery replacement. The IDs can be used by the controller to select the proper IR table from an IR database.

#### 6.5.8.1.2 Remote Storage of RF Statistics

The controller MAY acquire statistics about its link with the target and transfer these statistics to the target using the remote store procedure. The target can access these statistics for features that are beyond the scope of this document.

#### 6.5.8.1.3 Versioning

The controller SHALL be able to store the version numbers of its HW, SW, etc., on the target via the remote store procedure. The target can use this information for various purposes.

#### 6.5.8.1.4 Battery Status

The controller SHALL be able to store its battery status on the target via the remote store procedure. The target can use this information for various purposes.

#### 6.5.8.1.5 IR-RF Database

A database on the target device contains IR and RF codes that can be downloaded by the controller and mapped to specific keys. When IR codes have been retrieved, these keys can control legacy IR-based devices like TV, etc. When RF codes have been retrieved, associated keys can perform simultaneous IR and RF functions. For example, when the volume up button is pressed, the remote can send a unity gain RF code to the STB and volume up IR code to the TV.

The controller SHALL be able to retrieve IR codes for a subset of its keys from the IR database on the target.

The controller SHALL also be able to retrieve RF codes from the target database; these codes are sent together with the IR codes.

#### 6.5.8.1.6 Validation Configuration

The controller SHALL be able to retrieve validation configuration parameters from the target at the start of the validation procedure that it will use during the validation procedure. This allows the validation procedure user experience to be managed by the target without requiring a full firmware download.

#### 6.5.8.1.7 General Purpose Remote storage

The controller SHALL be able to store some general-purpose parameters on the target via the remote store procedure and retrieve them via the remote retrieve procedure after a battery replacement.

The interpretation of these general-purpose parameters is fully implementation-specific and therefore SHALL NOT be interpreted by the target.

### 6.5.8.2 Remote Information Base

As shown in Figure 13, each controller has its own section in the RIB and SHALL only be able to store and retrieve data in/from its own section.

The target application SHALL have full read/write access from/to all sections in the RIB and can be notified when the controller stores or retrieves data.

Each of these controller-specific sections SHALL be composed of a number of attributes listed in Table 23 and specified in more detail in the following paragraphs.

***Table 23 - RIB Attributes***

| RIB Attribute | Identifier | Index Inter-pretation | # Octets per Element | Writeable Flag | Description |
|---|---|---|---|---|---|
| Peripheral IDs | 0x00 | Device Type [4] | 4 | True | Identifiers of the Peripherals |
| RF Statistics | 0x01 | - | 16 | True | RF Statistics |

[4]It identifies the device type of the peripheral using the enumeration of device types specified in [RF4CE TYPE].

| RIB Attribute | Identifier | Index Inter-pretation | # Octets per Element | Writeable Flag | Description |
|---|---|---|---|---|---|
| Versioning | 0x02 | Part of the device (SW,HW etc.) | 4 | True | Versions of different parts of the device |
| Battery Status | 0x03 | - | 11 | True | Controller battery status information |
| Short RF Retry Period | 0x04 | - | 4 | False | The maximum time in µs a unicast acknowledged multichannel transmission shall be retried in case the Short RF Retry configuration is set. |
| IR-RF Database | 0xDB | Key | Variable | False | IR and RF codes for different keys |
| Validation Configuration | 0xDC | - | 6 | False | Configurable properties of the validation procedure |
| General Purpose | 0xFF | Row | 16 | True | General purpose remote storage |

Simple attributes are represented as a single element. More complex attributes SHALL be represented as a vector that is composed of smaller elements that are individually addressable using their index. This mechanism also allows the vector to be transferred in pieces due to the limited size of the RF frames.

An element spans an attribute-specific number of octets. However, the element size SHALL never be larger than *aplcMaxRIBAttributeSize* octets, since this is the maximal size of the Value field in the set attribute request and get attribute response command frames.

Some attributes cannot be modified by the controller, indicated by the Writeable flag. The controller SHALL only retrieve them from the RIB.

Since the target always has full read/write access from/to all attributes, it MAY set the value of attributes that have the Writeable flag set.

These attributes SHALL only be used by the controller to retrieve data that was stored there by the target.

The target SHALL NOT be required to implement all RIB attributes. The RIB attributes that are implemented are implementation-specific.

Note that the representation of the RIB in the target in Figure 13 is taken from the point of view of the controller and does not reflect the memory needs at the target side. Indeed the content of some attributes can be common for all controller sections, lowering the memory needs for the target.

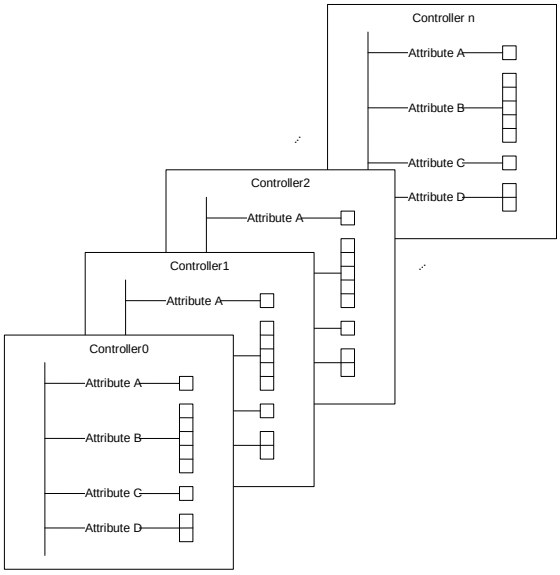


***Figure 13 - Remote Information Base on the Target***

**Note:** Each controller has its own RIB section spanning the different attributes.

#### *6.5.8.2.1 Peripheral IDs*

The Peripherals IDs attribute SHALL contain the identifiers for the Peripherals in the user's environment (e.g., brand and type of TV and/or other peripherals).

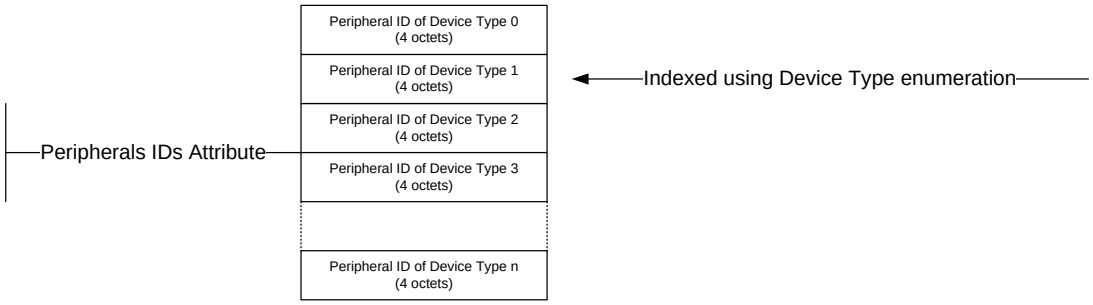


***Figure 14 - Data Structure of Peripherals IDs Attribute***

As shown in Figure 14, the data structure is a vector. The index SHALL specify the device type of the peripheral using the enumeration of device types specified in [RF4CE TYPE].

Each element of the data structure, i.e., each peripheral ID, SHALL be four (4) octets wide.

The Peripheral IDs attribute SHALL be read and written by the target, and can be remotely read and written by the controller.

The target MAY decide for which device types it supports this attribute, meaning this vector can be sparse.

#### *6.5.8.2.2 RF Statistics*

The RF Statistics attribute SHALL contain the RF statistics that the controller has acquired.

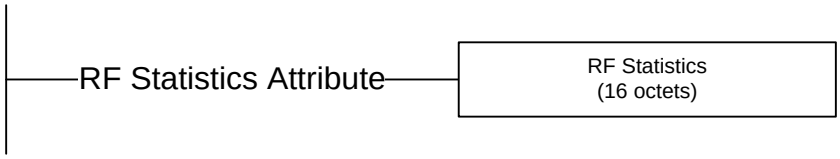


***Figure 15 - Data Structure of RF Statistics Attribute***

As shown in Figure 15, the RF Statistics attribute contains a single element. This element SHALL be 16 octets wide. The definition of the exact content of this attribute is beyond the scope of this document.

The RF Statistics attribute SHALL be read and written by the target, and can be remotely read or written by the controller. The target can decide if it supports this attribute.

### *6.5.8.3 Versioning*

The Versioning attribute SHALL contain the versions of the different parts of the controller.

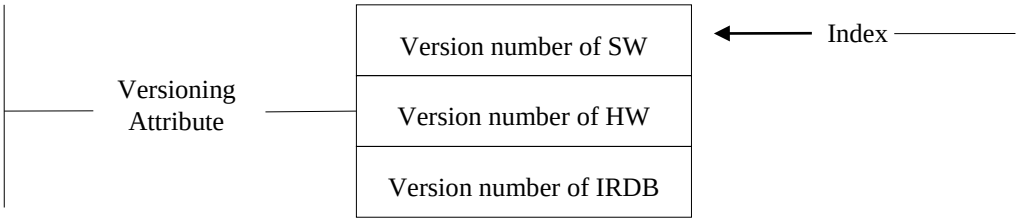


***Figure 16 - Data Structure of Versioning Attribute***

As shown in Figure 16, the data structure SHALL be a vector.

The index, specified in Table 24, SHALL indicate the part of the controller device the version number is referring to.

Each element of the data structure, i.e., each version number, SHALL be four (4) octets in length.

The Versioning elements SHALL be formatted as shown in Table 25 and Table 26.

***Table 24 - Indexing of the Versioning Attribute***

| Index | Part of controller device |
|---|---|
| 0x00 | Software |
| 0x01 | Hardware |
| 0x02 | IR Database |

***Table 25 - Format of the Software Versioning Elements***

| Byte 0 | Byte 1 | Byte 2 | Byte3 |
|---|---|---|---|
| Major | Minor | Revision | Patch |
| SW Version | | | |

***Table 26 - Format of the Hardware Versioning Elements***

| Byte 0 | | Byte 1 | Byte 2 | | Byte3 |
|---|---|---|---|---|---|
| **Bits 7-4** | **Bits 3-0** | **Bits 7-0** | **Bits 7-4** | **Bits 3-0** | **Bits 7-0** |
| Manu facturer | Model | HW Revision | Reserved | Lot Code (11-8) | Lot Code (7-0) |
| HW Version | | | | | |

***Table 27 - Format of the IRDB Versioning Elements***

| Byte 0 | Byte 1 | Byte 2 | Byte3 |
|---|---|---|---|
| Major | Minor | Revision | Patch |
| IRDB Version | | | |

#### *6.5.8.4 Battery Status*

The Battery Status attribute SHALL contain the battery status of the controller.

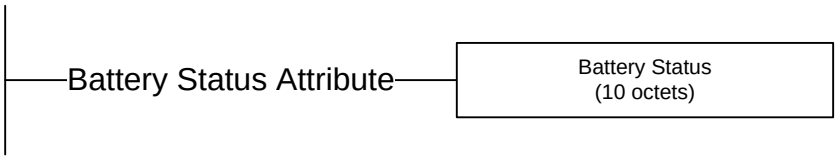


***Figure 17 - Data Structure of the Battery Status Attribute***

As shown in Figure 17, the Battery Status attribute SHALL contain a single element.

The element contained in the Battery Status attribute SHALL be 11 octets wide and be formatted as shown in Table 28.

***Table 28 - Format of the Battery Status Element***

| 1 Octet | 1 Octet | 4 Octet | 4 Octet | 1 Octet |
|---|---|---|---|---|
| Flags | Loaded Voltage Level | Number of RF codes transmitted | Number of IR codes transmitted | Unloaded Voltage Level |
| Battery Status | | | | |

#### *6.5.8.4.1 Flags*

The Flags field SHALL be one (1) octet in length and be formatted as shown in Table 30.

The flags SHALL be defined as specified in Table 29.

***Table 29 - Format of the Flags Field***

| Bit 0 | Bit 1 | Bit 2 | Bit 3-7 |
|---|---|---|---|
| Battery Replacement | Battery Charging | Impending Doom | Reserved |
| Flags | | | |

***Table 30 - Description of the Flags***

| Flag | Description |
|---|---|
| Battery Replacement | The battery is replaced. |
| Battery Charging | The battery is being charged. |
| Impending Doom | The battery has reached a critical level and the controller might no longer operate correctly. |

#### *6.5.8.4.2 Loaded Voltage Level*

The Loaded Voltage Level field SHALL be one (1) octet in length.

The Loaded Voltage Level field SHALL contain the voltage level reported by the controller under a load of 25 mA.

The Voltage Level value SHALL interpolate linearly between 0 V (encoded as 0x00) and 4 V (encoded as 0xFF).

#### *6.5.8.4.3 Number of RF Codes Transmitted*

The Number of RF Codes Transmitted field SHALL be four (4) octets in length.

The Number of RF Codes Transmitted field SHALL contain the number of RF code transmissions since the last battery replacement.

#### *6.5.8.4.4 Number of IR Codes Transmitted*

The Number of IR Codes Transmitted field SHALL be four (4) octets in length.

The Number of IR Codes Transmitted field SHALL contain the number of IR code transmissions since the last battery replacement.

#### *6.5.8.4.5 Unloaded Voltage Level*

The Unloaded Voltage Level field SHALL be one (1) octet in length.

The Unloaded Voltage Level field SHALL contain the voltage level reported by the controller under a load of 5 mA.

The Unloaded Voltage Level field SHALL interpolate linearly between 0 V (encoded as 0x00) and 4 V (encoded as 0xFF).

#### 6.5.8.5 Short RF Retry Period

The Short RF Retry Period attribute SHALL contain the maximum time in µs a unicast acknowledged multichannel transmission SHALL be retried if the Short RF Retry configuration is set.

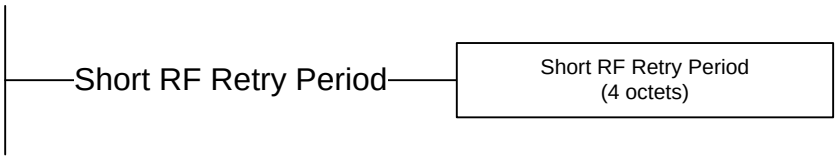


***Figure 18 -Short RF Retry Period Attribute***

As shown in Figure 18, the Short RF Retry Period attribute SHALL be formatted as a single 4-octet wide element.

The default value of the Short RF Retry Period SHALL be 100,000 µs = 100 ms.

#### 6.5.8.6 IR-RF Database

The IR-RF Database attribute SHALL contain the IR-RF codes for the keys on the controller.

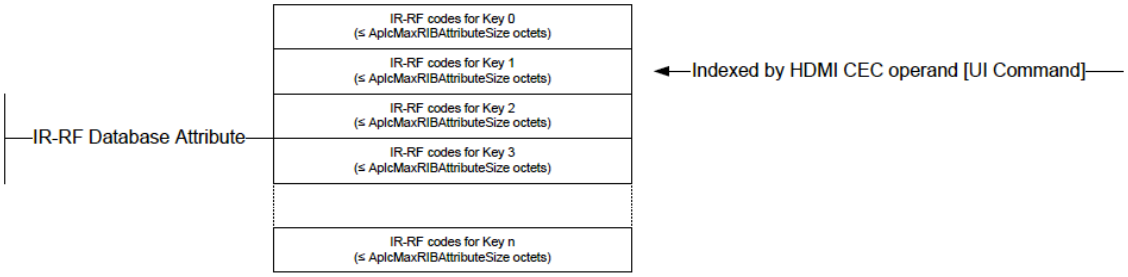


***Figure 19 - Data Structure of IR-RF Database Attribute***

As shown in Figure 19, the data structure SHALL be a vector.

The index SHALL specify the key using the HDMI [HDMI 1.4] CEC operand [UI Command].

Each element of the data structure, i.e., each key, SHALL have a variable width depending on the length of the IR and/or RF codes associated with the key.

Each element of the data structure SHALL be formatted as specified in Table 31.

The attribute SHALL be read and written by the target, but can only be remotely read by the controller.

The target SHALL decide for which keys it supports this attribute, which means that this vector can be sparse.

***Table 31 - Format of the IR-RF Database Elements***

| 1 Octet | 0 or Variable | 0 or Variable | 0 or Variable | 0 or Variable |
|---|---|---|---|---|
| Flags | RF Pressed Descriptor | RF Repeated Descriptor | RF Released Descriptor | IR Descriptor |
| IR-RF Database Attribute | | | | |

#### 6.5.8.6.1 Flags

The Flags field SHALL be one (1) octet in length.

The Flags field SHALL be formatted as shown in Table 32.

The Flags SHALL be defined as specified in Table 33.

***Table 32 - Format of the Flags Field***

| Bit 0 | Bits 1 | Bit 2 | Bit 3 | Bits 4-5 | Bit 6 | Bit 7 |
|---|---|---|---|---|---|---|
| RF Pressed Specified | RF Repeated Specified | RF Released Specified | IR Specified | Reserved | Use Default | Permanent |
| Flags | | | | | | |

***Table 33 - Flag Definitions***

| Flag | Description |
|---|---|
| Permanent | Indicates that the codes are permanent and can be used for all further presses of this key. |
| Use Default | Indicates that the default (known by the RC) RF and IR codes should be used. |
| IR Specified | Indicates that an IR descriptor is included in this attribute, and that an IR message should be generated when this key is pressed and kept pressed.<br>If Use Default is set, this field should be ignored (treated as if it was zero). |
| RF Pressed Specified | Indicates that an RF pressed descriptor is included in this attribute, and that an RF message should be generated when this key is pressed.<br>If Use Default is set, this field should be ignored (treated as if it was zero). |
| RF Repeated Specified | Indicates that an RF repeated descriptor is included in this attribute, and that an RF message should be generated when this key is kept pressed.<br>If Use Default is set, this field should be ignored (treated as if it was zero). |
| RF Released Specified | Indicates that an RF pressed descriptor is included in this attribute, and that an RF message should be generated when this key is released.<br>If Use Default is set, this field should be ignored (treated as if it was zero). |

#### 6.5.8.6.2 RF Pressed/Repeated/Released Descriptor

The RF Pressed/Repeated/Released Descriptor field SHALL have a variable length.

The RF Pressed/Repeated/Released Descriptor field SHALL only be included if the corresponding bit is set in the Flags field.

The RF Pressed/Repeated/Released Descriptor field SHALL be formatted as shown in Table 34.

***Table 34 - Format of RF Pressed/Repeated/Released Descriptor Field***

| 1 Octets | 1 Octets | 1 Octet | Variable |
|---|---|---|---|
| RF Config | RF4CE Tx Options | PayloadLength | RF4CE-NWK Payload |
| RF Pressed/Repeated/Released Descriptor | | | |

***Table 35 - Format of the RF Config Sub-field***

| Bit 0-3 | Bit 4 | Bit 5 | Bit 6-7 |
|---|---|---|---|
| Minimum number of transmissions | Keep Transmitting Until Key Release | Short Retry | Reserved |
| RF Config | | | |

***Table 36 - Description of RF Config Fields***

| Field | Description |
|---|---|
| Minimum number of transmissions | Indicates the minimum number of transmissions for this code. For acknowledged RF transmissions, this field is set to '1'. |
| Keep Transmitting Until Key Release | Indicates if the code should continue being transmitted after the minimum number of transmissions have taken place, when the key is kept pressed. (This field only applies to RF Repeated frames) |
| Short Retry | Indicates if the RF4CE retry period for UAM messages should be shorted for this code to increase responsiveness of the system. |

#### *6.5.8.6.3 IR Descriptor*

The IR Descriptor field SHALL have a variable length.

The IR Descriptor field SHALL only be included if the corresponding bit is set in the Flags field.

The IR Descriptor field SHALL be formatted as shown in Table 37.

***Table 37 - Format of IR Descriptor***

| 1 Octet | 1 Octet | Variable |
|---|---|---|
| IR Config | IR Code Length | IR Code |
| IR Descriptor | | |

***Table 38 - Format of IR Config Sub-field***

| Bit 0-3 | Bit 4 | Bit 5 | Bit 6 | Bit 7 |
|---|---|---|---|---|
| Minimum number of transmissions | Keep Transmitting Until Key Release | Reserved | Tweak Database | Reserved |
| IR Config | | | | |

***Table 39 - Description of IR Config Fields***

| Field | Description |
|---|---|
| Minimum number of transmissions | Indicates the minimum number of transmissions for the IR repeat frames. Only valid when Tweak Database is set to '1', otherwise follow behavior as defined by database. Special case: when Tweak Database is set to '1', setting this field to 0xF enforces the use of the value from the database. |

| Field | Description |
|---|---|
| Keep Transmitting Until Key Release | Indicates if the repeat frames should continue being transmitted after the minimum number of transmissions as defined by the database have been performed, in case the key remains pressed. Only valid when Tweak Database is set to '1', otherwise follow behavior as defined by database. |
| Tweak Database | Indicates that database behavior should be tweaked, using the "Minimum number of transmissions" and "Keep Transmitting Until Key Release" fields. |

#### *6.5.8.7 Validation Configuration*

The validation configuration attribute SHALL contain the configurable properties of the validation procedure.

At the start of the validation procedure, a controller MAY request these properties to use a more configurable validation procedure.

If the target doesn't support the Validation Configuration RIB or if the controller doesn't request these properties, the validation procedure SHOULD use the default settings.

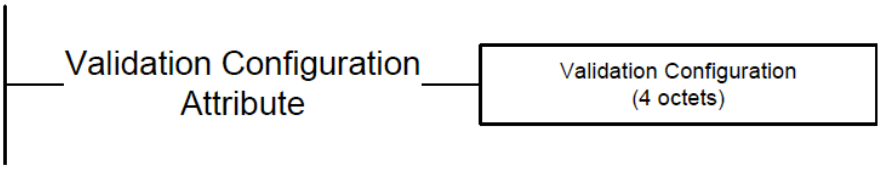


***Figure 20 - Data Structure of Validation Configuration Attribute***

As shown in Figure 20, the Validation Configuration attribute SHALL contain one (1) element.

The Validation Configuration element SHALL be four (4) octets wide.

The Validation Configuration element SHALL be formatted as shown in Table 40.

***Table 40 - Format of the Validation Configuration Element***

| 2 Octets | 2 Octets |
|---|---|
| Auto Check Validation Period | Link Lost Wait Time |
| Validation Configuration | |

##### *6.5.8.7.1 Auto Check Validation Period*

The Auto Check Validation Period field SHALL be:

- Two (2) octets in length
- Specified in milliseconds

##### *6.5.8.7.2 Link Lost Wait Time*

The Link Lost Wait Time field SHALL be:

- Two (2) octets in length
- Specified in milliseconds

#### *6.5.8.8 General Purpose*

The General Purpose attribute SHALL offer general-purpose storage for the controller.

As for the other attributes, this storage SHALL be per controller, since each controller has its own section.

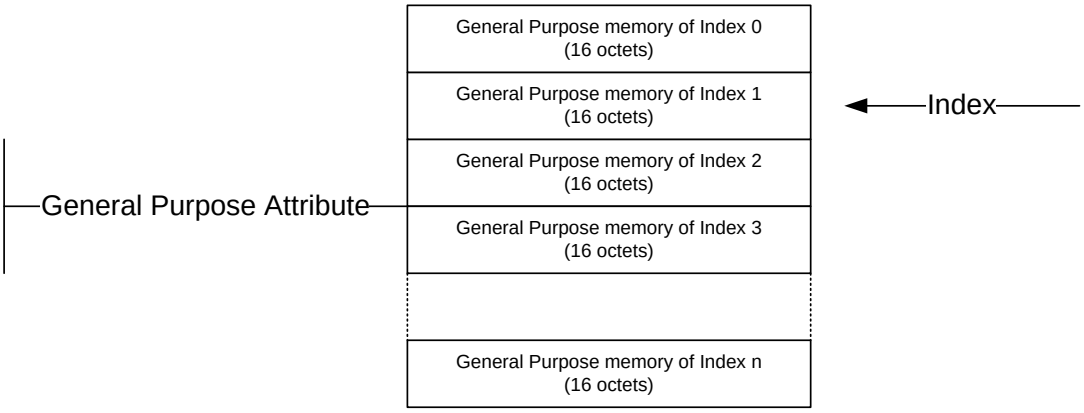


***Figure 21 - Data Structure of General Purpose Attribute***

As shown in Figure 21, the data structure is a vector. The index specifies the element. Each element of the data structure SHALL be 16 octets wide.

The General Purpose attribute SHALL be readable and writeable by the target.

The General Purpose attribute SHALL be readable and writeable remotely by the controller.

The target MAY decide if it supports this attribute and for how many elements.

The supported elements SHALL have a contiguous index range starting at zero, meaning the vector cannot be sparse.

##### *6.5.8.8.1 Remotely Storing Data in the RIB*

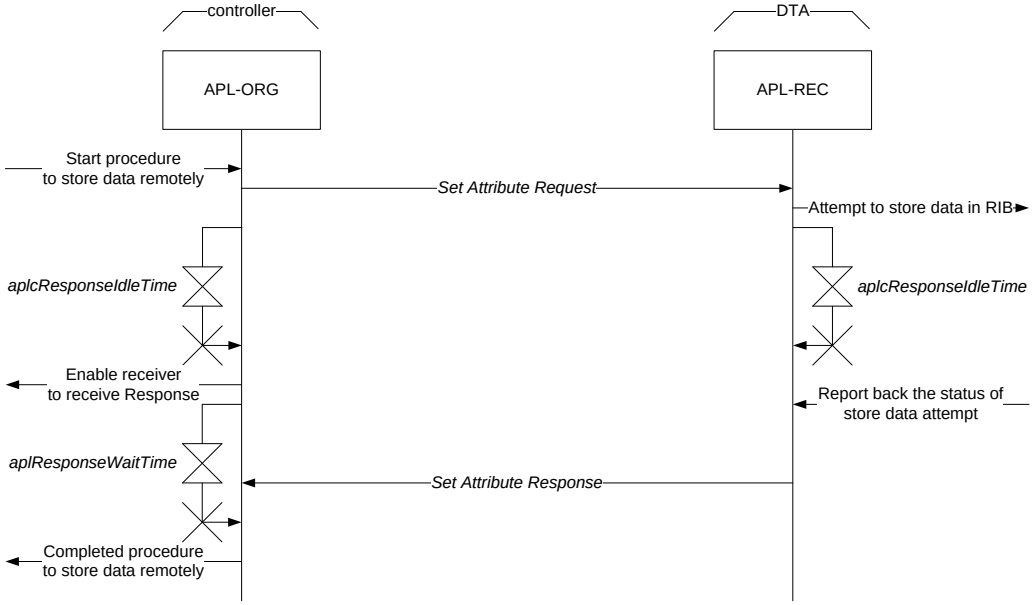


***Figure 22 - Message Sequence Chart for Remote Store Procedure***

Figure 22 illustrates the procedure that the controller SHALL use to remotely store data in the RIB of the target.

The controller SHALL transmit a set attribute request command frame to the target containing the data to be stored.

The target SHALL attempt to store this data in the RIB and report the result of this attempt.

#### *6.5.8.8.2 Controller Side Remotely Storing Data in the RIB*

When a controller device decides to store data remotely in the RIB of the target, it SHALL generate and transmit a set attribute request command frame to the target.

If the transmission was not successful, the remote store procedure SHALL fail.

If the transmission was successful, the controller SHALL wait *aplcResponseIdleTime* with its receiver disabled.

Next it enables its receiver and waits *aplResponseWaitTime* symbols for the corresponding set attribute response command frame to arrive.

If the controller receives the set attribute response command frame and the status is successful, the remote store procedure SHALL succeed.

If the controller receives the set attribute response command frame and the status is not successful, the remote store procedure SHALL fail.

If the controller doesn't receive the set attribute response command frame within the *aplResponseWaitTime* interval, the remote store procedure SHALL fail.

The receiver MAY be disabled again when the procedure is complete.

#### *6.5.8.8.3 Target Side Remotely Storing Data in the RIB*

In response to a set attribute request, the target SHALL wait *aplcResponseIdleTime*.

During the *aplcResponseIdleTime* interval, the target SHALL attempt to store the data in the RIB.

After the *aplcResponseIdleTime* interval, the target SHALL generate and transmit a set attribute response command to report to the controller the result of the attempt to store data in the RIB.

The result in the Status field of the set attribute response SHALL have following values:

- If the target was able to store the data in the RIB, the Status field is SUCCESS.
- If the set attribute request was issued with the identifier of a RIB attribute that is not supported by the target, the Status field is UNSUPPORTED_ATTRIBUTE.
- If the set attribute request was issued with an index that was out of range, the Status field is INVALID_INDEX.
- If the set attribute request was issued with a value that is not supported, out of range, or has an invalid length, the Status field is INVALID_PARAMETER.

#### 6.5.8.9 Remotely Retrieving Data from the RIB

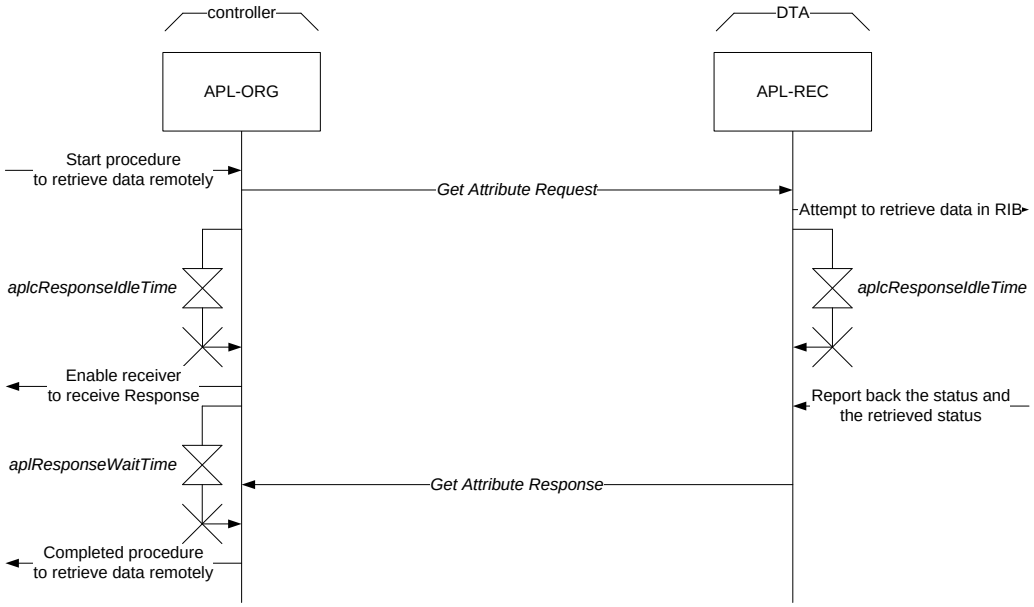


***Figure 23 - Message Sequence Chart for Remote Retrieve Procedure***

Figure 23 illustrates the procedure that a controller SHALL use to remotely retrieve data from the RIB of the target.

The controller SHALL transmit a get attribute request command frame to the target during the remote retrieval procedure.

The target SHALL attempt to retrieve the data in the RIB and report back the status of this attempt together with the retrieved data.

##### 6.5.8.9.1 Controller Side Remotely Retrieving Data in the RIB

When a controller device decides to retrieve/data remotely in the RIB of the target, it SHALL generate and transmit a get attribute request command frame to the target.

If the transmission was not successful, the remote retrieve procedure SHALL fail.

If the transmission was successful, the controller SHALL wait *aplcResponseIdleTime* with its receiver disabled.

After the *aplcResponseIdleTime* interval, the controller SHALL enable its receiver and wait *aplResponseWaitTime* symbols for the corresponding get attribute response command frame to arrive.

If the controller receives the get attribute response command frame and the status is successful, it can read the retrieved data from this command frame and the remote retrieve procedure SHALL succeed.

If the controller receives the get attribute response command frame and the status is not successful, the remote retrieve procedure SHALL fail.

If the controller does not receive the get attribute response command frame within the *aplResponseWaitTime interval*, the remote retrieve procedure SHALL fail.

The receiver MAY be disabled again when the procedure is complete.

##### 6.5.8.9.2 Target Side Remotely Retrieving Data in the RIB

In response to a get attribute request, the target SHALL wait *aplcResponseIdleTime*.

During this time the target SHALL attempt to retrieve the requested data from the RIB.

After the *aplcResponseIdleTime* interval, the target SHALL generate and transmit a get attribute response command frame to report to the controller the status of the attempt to retrieve data in the RIB, together with the retrieved data in case the attempt was successful.

The result in the Status field of the get attribute response SHALL have the following values:

- If the target was able to retrieve the data in the RIB, the Status field SHALL be SUCCESS.
- If the get attribute request was issued with the identifier of a RIB attribute that is not supported by the target, the Status field SHALL be UNSUPPORTED_ATTRIBUTE.
- If the get attribute request was issued with an index that was out of range, the Status field SHALL be INVALID_INDEX.

# Annex A RF4CE MSO PROFILE RF KEY CODES

This section provides RF key codes for the RF4CE MSO Profile.

Table 41 provides RF key codes used in conjunction with the RF4CE MSO Profile.

***Table 41 - MSO Key Codes***

| Remote Key or Function | RF4CE MSO Profile Code |
|---|---|
| TV POWER | 0x40 |
| MUTE | 0x43 |
| ALL POWER | Off: 0x6C |
| | On: 0x6D |
| VOL+ | 0x41 |
| VOL– | 0x42 |
| REPLAY | 0x4C |
| CH+ | 0x30 |
| CH– | 0x31 |
| REWIND | 0x48 |
| PLAY/PAUSE | 0x61 |
| FAST FORWARD | 0x49 |
| EXIT | 0x0D |
| MENU | 0x09 |
| RECORD | 0x47 |
| GUIDE | 0x53 |
| PAGE UP | 0x37 |
| UP ARROW | 0x01 |
| LEFT ARROW | 0x03 |
| OK | 0x00 |
| RIGHT ARROW | 0x04 |
| DOWN ARROW | 0x02 |
| LAST | 0x32 |
| INFO | 0x35 |
| PAGE DOWN | 0x38 |
| OCAP A (yellow triangle) / Function Key 3 | 0x74 |
| OCAP B (blue square) / Function Key 2 | 0x71 |
| OCAP C (red circle) / Function Key 0 | 0x72 |
| OCAP D (green diamond) / Function Key 1 | 0x73 |
| Digit 0 | 0x20 |
| Digit 1 | 0x21 |

| Remote Key or Function | RF4CE MSO Profile Code |
|---|---|
| Digit 2 | 0x22 |
| Digit 3 | 0x23 |
| Digit 4 | 0x24 |
| Digit 5 | 0x25 |
| Digit 6 | 0x26 |
| Digit 7 | 0x27 |
| Digit 8 | 0x28 |
| Digit 9 | 0x29 |
| 30-Second Skip Ahead | 0x4B |
| Input/Select | 0x34 |
| HOME | 0x10 |
| PROFILE | 0xA0 |
| CALL | 0xA1 |
| HOLD | 0xA2 |
| END | 0xA3 |
| VIEWS | 0xA4 |
| SELF-VIEW | 0xA5 |
| ZOOM IN | 0xA6 |
| ZOOM OUT | 0xA7 |
| MUTE MIC | 0x65 |
| STOP VIDEO | 0x64 |
| A | 0xB0 |
| B | 0xB1 |
| C | 0xB2 |
| D | 0xB3 |
| E | 0xB4 |
| F | 0xB5 |
| G | 0xB6 |
| H | 0xB7 |
| I | 0xB8 |
| J | 0xB9 |
| K | 0xBA |
| L | 0XBB |
| M | 0XBC |
| N | 0XBD |
| O | 0XBE |
| P | 0XBF |
| Q | 0XC0 |

| Remote Key or Function | RF4CE MSO Profile Code |
|---|---|
| R | 0XC1 |
| S | 0XC2 |
| T | 0XC3 |
| U | 0XC4 |
| V | 0XC5 |
| W | 0xC6 |
| X | 0XC7 |
| Y | 0xC8 |
| Z | 0XC9 |
| a | 0xCA |
| b | 0xCB |
| c | 0xCC |
| d | 0xCD |
| e | 0xCE |
| f | 0xCF |
| g | 0xD0 |
| h | 0xD1 |
| i | 0xD2 |
| j | 0xD3 |
| k | 0xD4 |
| l | 0xD5 |
| m | 0xD6 |
| n | 0xD7 |
| o | 0xD8 |
| p | 0xD9 |
| q | 0xDA |
| r | 0xDB |
| s | 0xDC |
| t | 0xDD |
| u | 0xDE |
| v | 0xDF |
| w | 0xE0 |
| x | 0xE1 |
| y | 0xE2 |
| z | 0xE3 |
| ? | 0xE4 |
| ! | 0xE5 |
| # | 0xE6 |

| Remote Key or Function | RF4CE MSO Profile Code |
|---|---|
| $ | 0xE7 |
| % | 0xE8 |
| & | 0xE9 |
| * | 0xEA |
| ( | 0xEB |
| ) | 0xEC |
| + | 0xED |
| - | 0xEE |
| = | 0xEF |
| / | 0xF0 |
| _ | 0xF1 |
| " | 0xF2 |
| : | 0XF3 |
| ; | 0xF4 |
| @ | 0xF5 |
| . | 0x2A |
| ' | 0xF6 |
| , | 0xF7 |
| BACKSPACE | 0xA8 |
| RETURN | 0x2B |
| LOCK/UNLOCK | 0xA9 |
| CAPS | 0xAA |
| ALT | 0xAB |
| SPACE | 0xAC |
| www. | 0xAD |
| .com | 0xAE |
| On Demand | 0xF8 |
| RF Bypass | 0xF9 |
| Next Favorite Channel | 0xFA |

# Appendix I Acknowledgements

CableLabs wishes to thank Comcast for development and contribution of this document to the cable industry.