ETSI EN 303 095 V1.3.1 (2018-05)

Reconfigurable Radio Systems (RRS);
Radio reconfiguration related architecture
for Mobile Devices (MD)

Reference
REN/RRS-0216

Keywords
architecture, mobile, SDR

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This European Standard (EN) has been produced by ETSI Technical Committee Reconfigurable Radio Systems (RRS).

National transposition dates	
Date of adoption of this EN:	17 May 2018
Date of latest announcement of this EN (doa):	31 August 2018
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	28 February 2019
Date of withdrawal of any conflicting National Standard (dow):	28 February 2019

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The scope of the present document is to define the radio reconfiguration related architecture for reconfigurable Mobile Devices. The work will be based on the system requirements defined in ETSI EN 302 969 [1] and the Use Cases defined in ETSI TR 103 062 [i.1] and ETSI TR 102 944 [i.2].

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI EN 302 969 (V1.3.1): "Reconfigurable Radio Systems (RRS); Radio Reconfiguration related requirements for Mobile Devices".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 103 062: "Reconfigurable Radio Systems (RRS) Use Cases and Scenarios for Software Defined Radio (SDR) Reference Architecture for Mobile Device".

[i.2] ETSI TR 102 944: "Reconfigurable Radio Systems (RRS); Use Cases for Baseband Interfaces for Unified Radio Applications of Mobile Device".

[i.3] Recommendation ITU-T M.60: "Maintenance Terminology and Definitions".

[i.4] ETSI TS 103 436: "Reconfigurable Radio Systems (RRS); Security requirements for reconfigurable radios".

[i.5] ETSI TR 103 087: "Reconfigurable Radio Systems (RRS); Security related use cases and threats in Reconfigurable Radio Systems".

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Application Processor (AP): part of mobile device hardware working under OS control and on which User Applications, among others, are executed

Baseband Parameter Aggregation (BPA): unit collecting all the context information to be transferred to the monitor

NOTE: The BPA unit converts the context information into metric(s) such that a minimum bandwidth is consumed during the procedure of transferring the context information to the monitor. Those metrics may include Received Signal Strength Indication (RSSI) measurement, multi-RAT performance metrics, etc.

communication services layer: layer related to communication services supporting generic applications

NOTE: A communication services layer supports generic applications like Internet access. In the present document, it consists of Administrator, Mobility Policy Manager (MPM), Networking stack and Monitor.

configcodes: result of compiling the source codes of a Radio Application (RA), which is either configuration codes of Radio Virtual Machine (RVM) or executable codes for a particular target platform

NOTE: In the case when RA provider makes a high level code based on a target platform, a result of compiling RA source codes is configcodes which is executable on the target platform. In the other case, when RA provider makes a high level code without considering a target platform, a result of front-end compiling of RA source codes is an Intermediate Representation (IR) which should be back-end compiled for operating on a specific target platform.

data flow: logical channel between Flow Controller (FC) and a Unified Radio Applications (URA) created by FC to send to or receive data elements (octets, packets or other granularity) from URA

environmental information: set of values that can affect the execution of RAs on a Radio Computer

NOTE: Environmental information consists of information related to the execution of RA(s), such as Buffer Overflow, Resource Allocation, etc.

Functional Block (FB): function needed for real-time implementation of RA(s)

NOTE 1: A functional block includes not only the modem functions in Layer1 (L1), Layer2 (L2), and Layer 3 (L3) but also all the control functions that should be processed in real-time for implementing given RA(s).

NOTE 2: Functional blocks are categorized into Standard Functional Blocks (SFBs) and User Defined Functional Blocks (UDFBs). In more details:

1) *SFB* can be shared by many RAs. For example, Forward Error Correction (FEC), Fast Fourier Transform (FFT)/Inverse Fast Fourier Transform (IFFT), (de)interleaver, Turbo coding, Viterbi coding, Multiple Input Multiple Output (MIMO), Beamforming, etc. are the typical category of standard functional block.

2) *UDFB* include those functional blocks that are dependent upon a specific RA. They are used to support special function(s) required in a specific RA or to support a special algorithm used for performance improvement. In addition, a user defined functional block can be used as a baseband controller functional block which controls the functional blocks operating in baseband processor in real-time and to control some context information processed in real-time.

NOTE 3: Each functional block has its unique name, Input, Output, and properties.

peer equipment: any communication counterpart of a reconfigurable radio equipment

NOTE: The peer equipment can be reached by establishing a (logical) communications link (i.e. an association) between the reconfigurable radio equipment and peer equipment. Examples of peer equipment include Wide Local Area Network (WLAN) access points, Internet Protocol (IP) access nodes, etc.

Radio Application (RA): software which enforces the generation of the transmit RF signals or the decoding of the receive RF signals

NOTE 1: The software is executed on a particular radio platform or an RVM as part of the radio platform.

NOTE 2: RAs might have different forms of representation. They are represented as:

- Source codes including Radio Library calls of Radio Library native implementation and Radio HAL calls.
- IRs including Radio Library calls of Radio Library native implementation and radio HAL calls.
- Executable codes for a particular radio platform.

radio computer: part of mobile device hardware working under ROS control and on which RAs are executed

NOTE: A Radio Computer typically include programmable processors, hardware accelerators, peripherals, etc. RF part is considered to be part of peripherals.

Radio Control Framework (RCF): control framework which, as a part of the OS, extends OS capabilities in terms of radio resource management

NOTE: RCF is a control framework which consists of Configuration Manager (CM), Radio Connection Manager (RCM), Flow Controller (FC) and Multiradio Controller (MRC). The Resource Manager (RM) is typically part of OS.

Radio Controller (RC): functional component of RA for transferring context information from corresponding RAs to monitor

NOTE: An RC, which may operate in an application processor in non real-time, accesses RAs which operates in Radio Computer in real time. The monitor, to which the context information is transferred using RC, provides context information to Adminstrator and/or Mobility Policy Manager (MPM) for application(s) to be performed using the context information, for example, terminal-centric configuration.

Radio Frequency Transceiver (RF Transceiver): part of radio Platform converting, for transmission, baseband signals into radio signals, and, for reception, radio signals into baseband signals

Radio Library (RL): library of SFB that is provided by a platform vendor in a form of platform-specific executable code

NOTE 1: SFBs implement reference codes of functions which are typical for radio signal processing. They are not atomic and their source codes are typed and visible for RA developers.

NOTE 2: An SFB is implemented through a Radio Hardware Abstraction Layer (HAL) when the SFB is implemented on hardware accelerators. Radio HAL is part of ROS.

Radio Operating System (ROS): any appropriate OS empowered by RCF

NOTE: ROS provides RCF capabilities as well as traditional management capabilities related to management of RP such as resource management, file system support, unified access to hardware resources, etc.

radio platform: part of mobile device hardware which relates to radio processing capability, including programmable components, hardware accelerators, RF transceiver, and antenna(s)

NOTE: A radio Platform is a piece of hardware capable of generating RF signals or receiving RF signals. By nature, it is heterogeneous hardware including different processing elements such as fixed accelerators, e.g. Application-Specific Integrated Circuit (ASIC), or reconfigurable accelerators, e.g. FPGAs, etc.

Radio Virtual Machine (RVM): abstract machine which supports reactive and concurrent executions

NOTE: An RVM may be implemented as a controlled execution environment which allows the selection of a trade-off between flexibility of base band code development and required (re-)certification efforts.

reconfigurable mobile device: mobile device with radio communication capabilities providing support for radio reconfiguration

NOTE: Reconfigurable mobile devices include but are not limited to: smartphones, feature phones, tablets, and laptops.

reference point: conceptual point at the conjunction of two non-overlapping functions that can be used to identify the type of information passing between these functions

NOTE: This definition is introduced by Recommendation ITU-T M.60 [i.3].

shadow radio platform: platform where configcodes can be directly executed when it corresponds to the target radio platform or, when it corresponds to an RVM, compiled and executed

NOTE: If the Shadow radio platform is equivalent to the target radio platform, then a front-end compiler will generate the executable code for the target radio platform and configcodes are equivalent to the executable code for that radio platform.

3.2 Symbols

For the purposes of the present document, the following symbols apply:

M_1 Number of SFBs implemented on Radio computer

M_2 Number of SFBs implemented on hardware accelerators

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AOT	Ahead-Of-Time
AP	Application Processor
ASF	Administrator Security Function
ASIC	Applications-Specific Integrated Circuit
BE	Back End
BPA	Baseband Parameter Aggregation
CM	Configuration Manager
CSL	Communication Services Layer
FC	Flow Controller
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FM	File Manager
FPGA	Field Programmable Gate Array
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GPS	Global Positioning System
HAL	Hardware Abstraction Layer
HW	HardWare
ID	Identification
IFFT	Inverse Fast Fourier Transform
IP	Internet Protocol
IR	Intermediate Representation
JIT	Just-In-Time
KMS	Key Management System
MAC	Medium Access Control
MD	Mobile Device
MDRC	Mobile Device Reconfiguration Class
MIMO	Multi-Input-Multi-Output
MPM	Mobility Policy Manager
MRC	MultiRadio Controller
MURI	MUltiRadio Interface

OEM	Original Equipment Manufacturer
OS	Operating System
RA	Radio Application
RAP	Radio Application Package
RAT	Radio Access Technology
RC	Radio Controller
RCF	Radio Control Framework
RCM	Radio Connection Manager
RF	Radio Frequency
RL	Radio Library
RM	Resource Manager
ROS	Radio Operating System
RPI	Radio Programming Interface
RRFI	Reconfigurable Radio Frequency Interface
RRS	Reconfigurable Radio Systems
RRS-CA	Reconfigurable Radio Systems Configuration Authority
RRS-CM	RRS Configuration Manager
RRS-CP	RRS Configuration Provider
RVM	Radio Virtual Machine
SDR	Software Defined Radio
SFB	Standard Functional Block
SW	SoftWare
TAD	Transfer of Authority Document
TX/RX	Transmission/Reception
UDFB	User Defined Functional Block
URA	Unified Radio Applications
URAI	Unified Radio Applications Interface
WLAN	Wireless Local Area Network

4 Architectural Reference Model for Reconfigurable Mobile Devices

4.1 Introduction

The present document describes those elements of a mobile device which are related to the software radio reconfiguration only. For this reason,the usage of the term "architecture" is limited to those elements and not to the overall HW/SW architecture of a mobile device which is out of the scope of the present document.

The present document is organized as follows:

Clause 4.2 describes the reconfigurable mobile device architecture in term of its components and entities.

Clause 4.3 describes the architecture reference model for multiradio applications.

Clause 4.4 describes the "Radio Computer".

Clause 4.5 describes the Radio Virtual Machine as part of the architecture.

Clause 4.6 describes the Unified Radio Application.

Clause 4.7 describes the security architecture for reconfigurable mobile devices.

Clause 5 describes the (logical) interfaces between the identified components/entities.

Clause 6 lists the operating procedures of a reconfigurable mobile devices.

Clause 4 includes a list of tables mapping the system requirements as defined in ETSI EN 302 969 [1] to the different entities/components/units which have been identified. In general, according to the MDRC [1] the reconfigurable mobile device belongs to, all the related mandatory functional requirements described in ETSI EN 302 969 [1] shall be implemented.

4.2 Reconfigurable Mobile Devices - Architecture Components for Radio Reconfiguration

4.2.1 High level description

Figure 4.1 shows the reconfigurable mobile device architectural components related to the radio reconfiguration as well as the related entities. As shown in the figure, the following components can be identified:

- Communication Services Layer (CSL):
  - 4 logical entities: Administration, Mobility Policy Manager, Networking Stack and Monitor.
- Radio Control Framework (RCF):
  - 5 logical entities: Configuration Manager, Radio Connection Manager, Multi-Radio Controller, Resource Manager and Flow Controller.
- Unified Radio Applications (URA).
- Radio Platform (consisting of RF Transceiver, Baseband, etc.).

These 4 components consist of Software (CSL, RCF) and/or Hardware (radio platform) entities and they shall be interconnected through well defined interfaces as follows:

- Multiradio Interface (MURI) between CSL and RCF.
- Unified Radio Application Interface (URAI) between RCF and URA.
- Reconfigurable Radio Frequency Interface (RRFI) between URA and RF Transceiver.

The above mentioned interfaces are not covered by the present document.

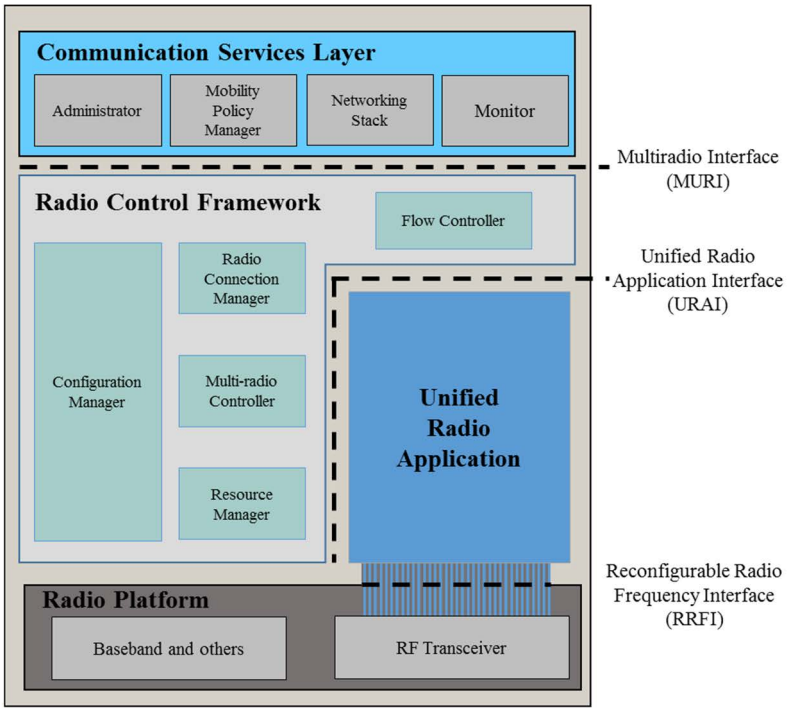

Figure 4.1: Reconfigurable mobile device architecture components for radio reconfiguration

For each component, the required entities depend on the MDRC [1]. A Reconfigurable Mobile Device shall support all the components and their entities as required by the corresponding MDRC as shown in Table 4.1. In case that a Reconfigurable Mobile Device supports multiple MDRCs, the concerned Reconfigurable Mobile Device shall support all the components and entities related to the highest supported MDRC.

Table 4.1: Required Components of the Reconfigurable Mobile Device Architecture in function of the Mobile Device Reconfiguration Class

Mobile Device Reconfiguration Class	Required CSL Entities	Required RCF Entities	Required Interfaces
MDRC-0	None	None	None
MDRC-1	Administrator, Mobility Policy Manager, Networking Stack, Monitor	Configuration Manager, Radio Connection Manager, Flow Controller	MURI
MDRC-2, MDRC-5	Administrator, Mobility Policy Manager, Networking Stack, Monitor	Configuration Manager, Radio Connection Manager, Multi-Radio Controller, Flow Controller	MURI, URAI, RRFI
MDRC-3, MDRC-6	Administrator, Mobility Policy Manager, Networking Stack, Monitor	Configuration Manager, Radio Connection Manager, Multi-Radio Controller, Flow Controller	MURI, URAI, RRFI
MDRC-4, MDRC-7	Administrator, Mobility Policy Manager, Networking Stack, Monitor	Configuration Manager, Radio Connection Manager, Multi-Radio Controller, Resource Manager, Flow Controller	MURI, URAI, RRFI

The following clauses describe in more details the identified components as well as the related logical entities.

4.2.2 Communication Services Layer (CSL)

The CSL is a layer related to communication services supporting both generic applications and specific applications related to multiradio applications. CSL includes the following 4 entities:

- **Administrator entity**

  The Administrator entity shall include at least functions to request installation or uninstallation of URA, and creating or deleting instances of URA. This typically includes the provision of information about the URA, their status, etc. Furthermore, the Administrator includes two sub-entities: the Administrator Security Function (ASF) and the RRS Configuration Manager (RRS-CM).

NOTE: In case that a snapshot function is required, the Administrator entity may store relevant RAPs, their configuration parameters and information on the URA installation and execution history. When required, the same steps can be executed by the Administrator entity to fall back to a previous snapshot.

- **Mobility Policy Manager (MPM) entity**

  The MPM shall include at least functions for monitoring of the radio environments and MD capabilities, to request activation or deactivation of URA, and to provide information about the URA list. It shall also make selection among different radio access technologies and discover peer communication equipment and arrangement of associations.

- **Networking stack entity**

  The Networking stack entity shall include at least functions for sending and receiving of user data.

- **Monitor entity**

  The Monitor entity shall include at least functions to transfer information from URA to user or proper destination entity in MD.

4.2.3 Radio Control Framework (RCF)

The RCF provides a generic environment for the execution of URA, and a uniform way of accessing the functionality of the Radio Computer and individual RAs. RCF provides services to CSL via the Multiradio Interface (MURI).

The RCF includes the following 5 entities for managing URA [i.2]:

- **Configuration Manager (CM) entity**

  The CM shall include at least functions for installing/uninstalling and creating/deleting instances of URA as well as management of and access to the radio parameters of the URA.

- **Radio Connection Manager (RCM) entity**

  The RCM shall include at least functions for activating/deactivating URA according to user requests, and to management of user data flows, which can also be switched from one RA to another.

- **Flow Controller (FC) entity**

  The FC shall include at least functions for sending and receiving of user data packets and controlling the flow of signalling packets.

- **Multiradio Controller (MRC) entity**

  The MRC shall include at least functions to schedule the requests for radio resources issued by concurrently executing URA, and to detect and manage the interoperability problems among the concurrently executed URA.

- **Resource Manager (RM) entity**

  The RM shall include at least functions to manage the computational resources, to share them among simultaneously active URA, and to guarantee their real-time execution.

4.2.4 Unified Radio Application (URA)

As described in clause 4.2.3, the RCF, which represents functionalities provided by the Radio Computer, requires all RAs to be subject to a common reconfiguration, multiradio execution and resource sharing strategy framework (depending on the concerned MDRC). Since all RAs exhibit a common behaviour from the reconfigurable MD perspective, those RAs are called URAs. The services relate to activation and deactivation, peer equipment discovery and maintenance of communication over user data flows are provided at Unified Radio Application Interface (URAI), which is an interface between URA and RCF.

4.2.5 Architectural Components System Requirements mapping

The logical entities above described are mapped to the system requirements described in ETSI EN 302 969 [1] as shown in Table 4.2.

Table 4.2: Mapping of Architectural Components to the system requirements described in ETSI EN 302 969 [1]

Entity/Component/Unit	System Requirements [1]	Comments
Administrator	R-FUNC-MDR-01, R-FUNC-MDR-02, R-FUNC-SEC-01, R-FUNC-SEC-02	The reconfigurable MD configuration is performed through downloading of the RAP into the reconfigurable MD and its installation. The requirements are described in clauses 6.4.1, 6.4.2, and 6.6 of ETSI EN 302 969 [1]
Mobility Policy Manager	R-FUNC-RAT–04, R-FUNC-MDR-03	RAP into the reconfigurable MD and its installation. The requirements are described in clauses 6.1.4 and 6.4.3 of ETSI EN 302 969 [1]
Networking stack Flow Control	R-FUNC-RA-04, R-FUNC-RAT–05	Management of data flows is required for basic TX/RX operation. The requirement is described in clause 6.2.4 of ETSI EN 302 969 [1]
Monitor	R-FUNC-RA-05	The RC in RA ensures the availability of context information. The requirement is described in clause 6.2.5 of ETSI EN 302 969 [1]
Configuration Manager Radio Connection Manager	R-FUNC-MDR-03	The radio configuration of a reconfigurable MD is realized with the activation of URA. The requirement is described in clause 6.4.3 of ETSI EN 302 969 [1]
Multiradio Controller	R-FUNC-RAT–01, R-FUNC-RAT–02, R-FUNC-RAT–03, R-FUNC-RAT–05, R-FUNC-RAT–06	The proposed Mobile Device Architecture is suitable to support Multiple (parallel) connections to (heterogeneous) RATs. The requirements are described in clauses 6.1.1 and 6.1.2 of ETSI EN 302 969 [1]
Resource Manager	R-FUNC-MDR-05	In case of dynamic resource sharing, the resource allocation is performed in run time. The requirements is described in clause 6.4.5 of ETSI EN 302 969 [1]

4.3 Reconfigurable Mobile Devices - Architecture Reference Model for Multiradio Applications

4.3.1 High level description

Figure 4.2 exemplifies a Reconfigurable MD architecture reference model for multiradio applications. As shown in the figure, the reconfigurable MD architecture shall include at least a Radio Computer. In the example of Figure 4.2, the red-dotted part belongs to either Radio Computer or Application Processor depending on the specific implementation.

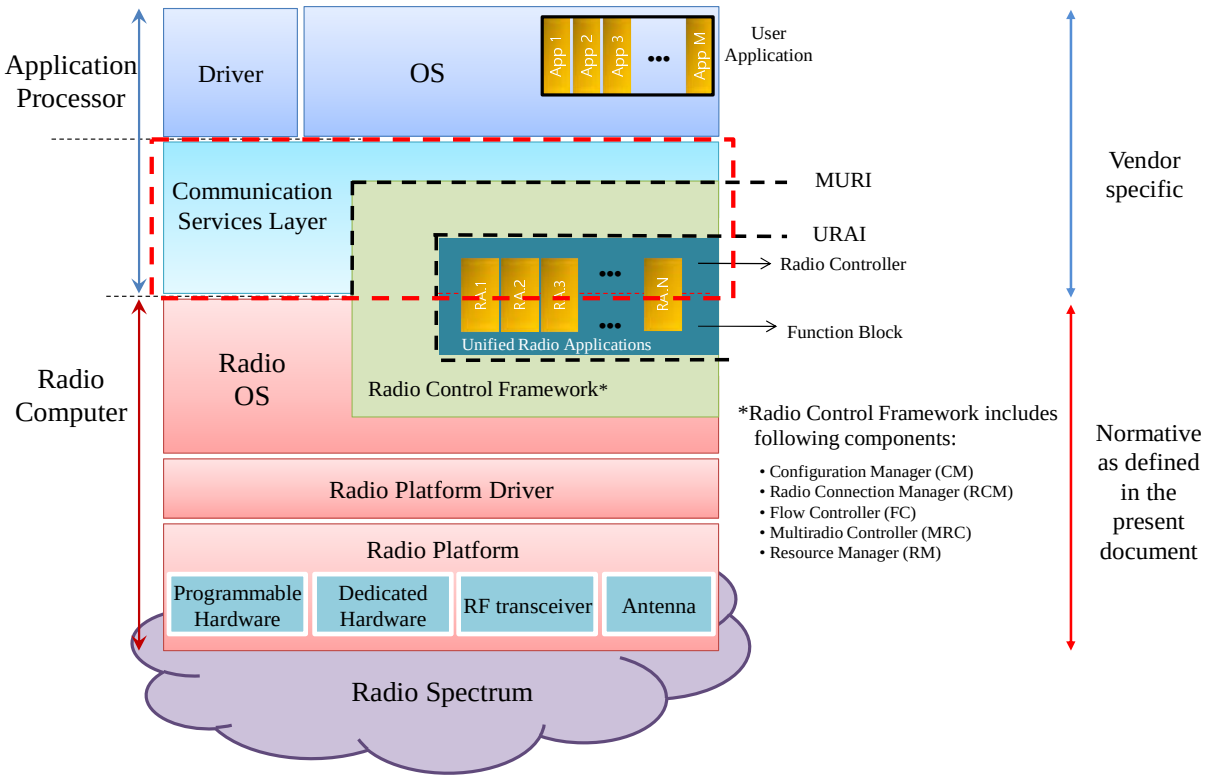

Figure 4.2: Reconfigurable Mobile Device (MD) architecture Reference Model for multiradio applications

In the example of Figure 4.2, the operation of Application Processor is performed by a given Operating System (OS), which is preferably performed on non-real-time bases, whereas Radio Computer's operation is performed by another OS, which should support real-time operations of URA. The OS of Radio Computer is referred to as Radio OS (ROS) in the present document.

The AP includes the following components:

- A Driver which has the purpose of activating the hardware devices (such as camera, speaker, etc.) on a given MD.
- A non-real time OS for execution of Administrator, MPM, Networking stack and Monitor which are part of the CSL as above described. For multiradio applications the OS may include RCF (Application Processor part).
- The Radio Controller (RC) in Radio Application (RA) sending context information to the Monitor and send/receive data to/from Networking stack.

The Radio Computer shall include the following components:

- ROS is a real-time Operating System.
- A radio platform driver which is a hardware driver for the ROS to interact with the radio platform hardware.
- The 5 entities of the RCF, specified in clause 4.2.3, are classified into two groups. One group relates to real-time execution and the other group to non-real-time execution as shown in Figure 4.2. Which entities of RCF interface relate to real-time and non-real-time execution, can be determined by each vendor.

4.3.2 Reference Model System Requirements mapping

The architecture reference model above described is mapped to the system requirements described in ETSI EN 302 969 [1] as shown in Table 4.3.

Table 4.3: Mapping of Reference Model to the system requirements described in ETSI EN 302 969 [1]

Entity/Component/Unit	System Requirements [1]	Comments
Application Processor (vendor specific)	R-FUNC-RA-05	The Radio Controller in RA ensures the availability of context information. The requirement is described in clause 6.2.5 of ETSI EN 302 969 [1]
Radio Computer	R-FUNC-MDR-09, R-FUNC-RA-06	ROS enables management of timing constraints and provides interface between URA and radio platform. The requirements are described in clauses 6.4.9 and 6.2.6 of ETSI EN 302 969 [1]

4.4 Reconfigurable Mobile Devices - Radio Computer

4.4.1 High level description

The System Architecture for a Radio Computer is illustrated in Figure 4.3 and Figure 4.4. Some of the entities included in the figures below may be located externally (in the "Cloud") in order to off-load processing from the concerned Mobile Devices. As example, the Back End compiler in Figure 4.3 is moved into the "Cloud" as illustrated in Figure 4.4.

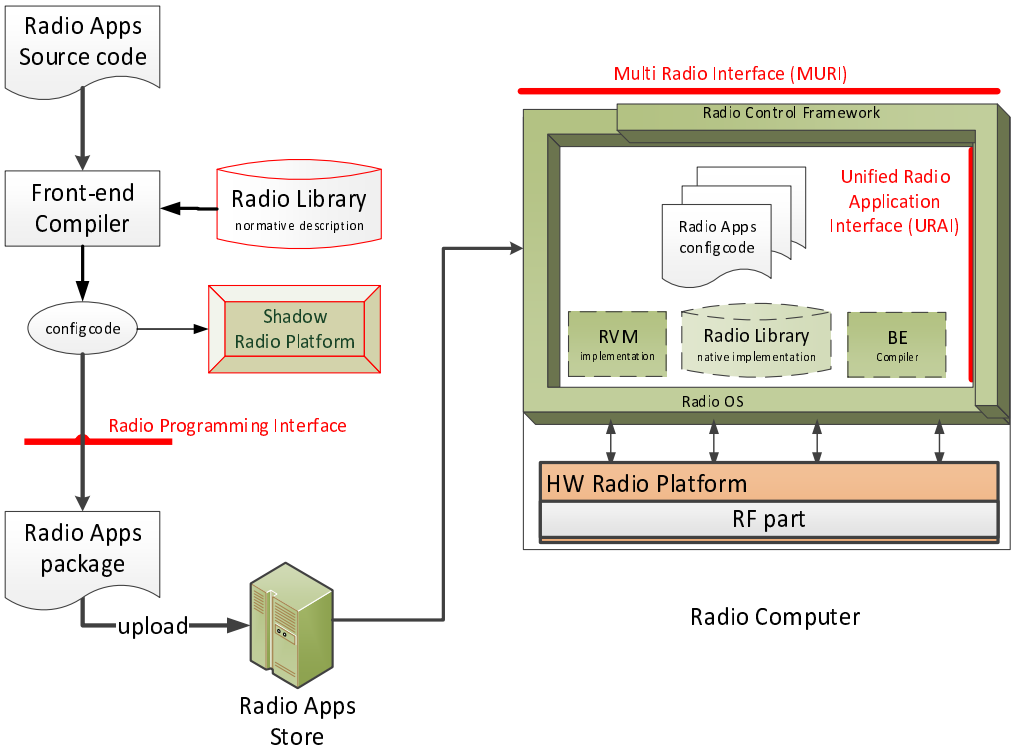

Figure 4.3: System architecture for Radio Computer where Radio Library and Back End (BE) compiler are included within the Radio Computer

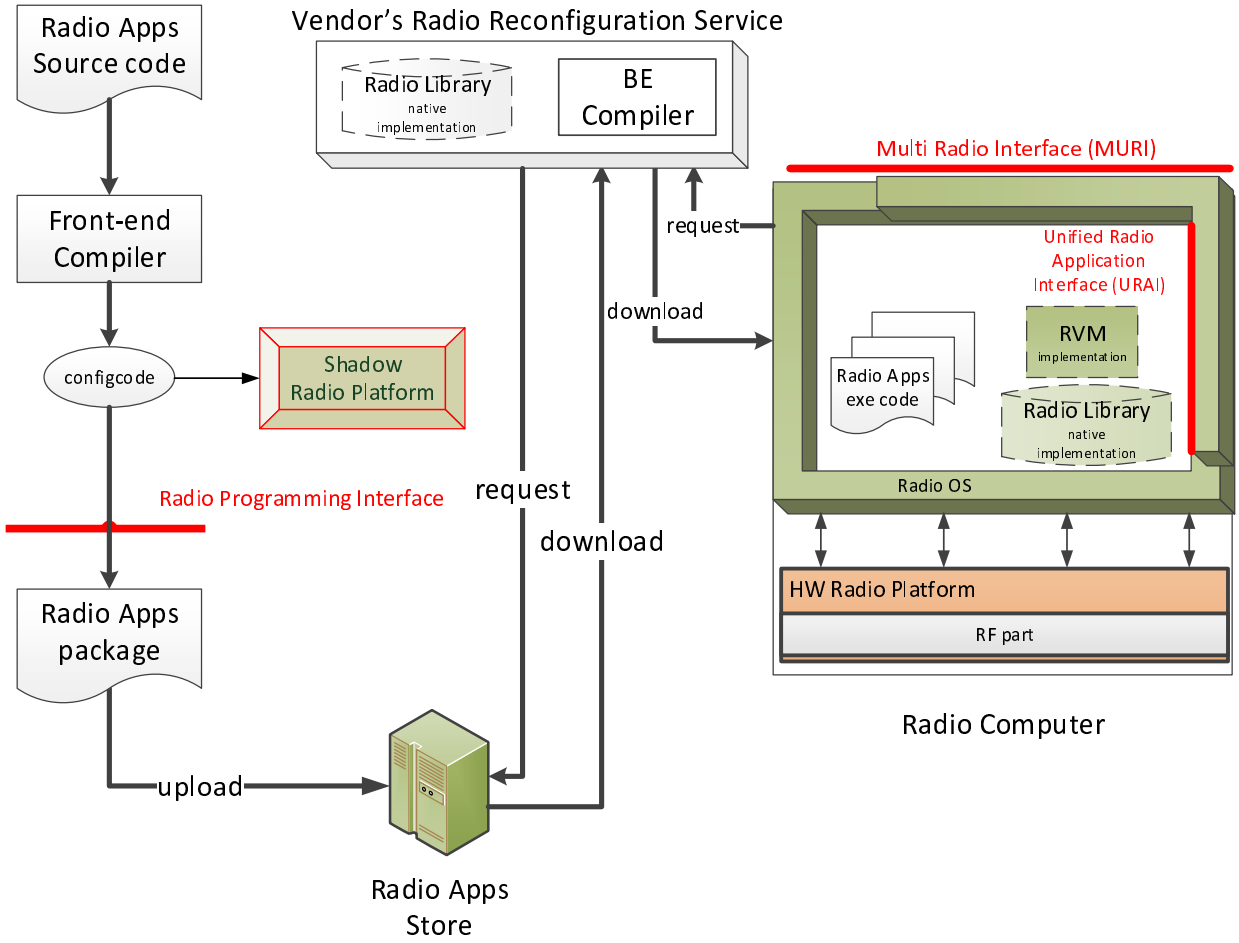

Figure 4.4: System architecture for Radio Computer where Radio Library and BE compiler are provided at a cloud outside the Radio Computer

Certification is required for Configcodes.

The Radio Computer shall provide communication capabilities for reconfigurable MDs and shall consist of:

- ROS including RCF.
- URA configuration codes (configcodes), which are:
  - Executable codes for MDRC-2. MDRC-3 and MDRC-4.
  - Source codes or IR for MDRC-5, MDRC-6 and MDRC-7.
- Radio Virtual Machine (RVM) for MDRC-5, MDRC-6 and MDRC-7.
- Radio Library Native Implementation for MDRC-5, MDRC-6 and MDRC-7 when URA configcodes is compiled within MD, or when URA configcodes is compiled in a cloud with dynamic linking.
- Radio platform.

URA Configcodes shall be executable codes for MDRC-2, MDRC-3 and MDRC-4, or shall be interpreted by the RVM for MDRC-5, MDRC-6 and MDRC-7.

For MDRC-2, MDRC-3 and MDRC-4, a front-end compiler shall generate the executable code for the target platform and configcodes are equivalent to the executable code for that target platform. Hence, Radio Library native implementation and Back-end Compiler shown with the dotted line in Figure 4.3 is not required in the Radio Computer.

The RVM (see also clause 4.4) is an Abstract Machine which is capable of executing configcodes and it is independent of the hardware. The implementation of an RVM is target Radio Computer specific and it includes the Back-end Compiler which might provide Just-in-Time (JIT) or Ahead-of-Time (AOT) method for compilation of configcodes into executable codes.

For MDRC-5, MDRC-6 and MDRC-7, where URA configcodes are source codes or IR, the Back-end Compiler can be implemented in 2 different ways as follows:

1) The URA configcodes are source codes or Intermediate Representation (IR) that is to be compiled at a given MD.

2) The URA configcodes are source codes or Intermediate Representation (IR) that is to be compiled at a Cloud.

In the former cases, as shown in Figure 4.3, Radio Library Native Implementation and Back-end Compiler are given in Radio Computer. Therefore, in this case, URA configcodes is downloaded into Radio Computer in the form of source code or IR and it is transformed into corresponding executable code through the Back-end Compiler within Radio Computer. Note that the Back-end Compiler can be a part of RVM in this case. In the latter case, as shown in Figure 4.4, the compilation process is performed at a cloud not within Radio Computer. Therefore, URA configcodes is downloaded into Radio Computer in the form of executable code as a result of the compilation at the cloud. It means that platform vendor should provide the Back-end Compiler and/or Radio Library Native Implementation at a cloud in accordance with their radio platform. Note that the Radio Library Native Implementation should be provided in a cloud/MD in the case of static/dynamic linking which will be explained in more detail in clause 4.6.

The Radio Library shall consist of Standard Functional Blocks (SFBs) representing the computational basis. An RA shall be expressed as a set of these interconnecting SFBs together with User Defined Functional Blocks (UDFBs) [1]. SFBs to be provided from the Radio Library normative description shall be represented in a platform-independent normative language. The native implementation of the Radio Library shall be provided as platform-specific codes of the SFBs from the library for the target platform. A Radio Library shall be extendable.

As illustrated in Figure 4.3 and Figure 4.4, the access to a RadioApp Store shall require an interface: the Radio Programming Interface (RPI). The definition of this interface is out of scope of the present document.

4.4.2 Radio Computer System Requirement Mapping

The radio computer above described is mapped to the system requirements described in ETSI EN 302 969 [1] as shown in Table 4.4.

Table 4.4: Mapping of Radio Computer to the system requirements described in ETSI EN 302 969 [1]

Entity/Component/Unit	System Requirements [1]	Comments
Radio Computer Architecture	R-FUNC-MDR-04, R-FUNC-MDR-10, R-FUNC-MDR-13	The requirements are described in clauses 6.4.4, 6.4.10 and 6.4.13 of ETSI EN 302 969 [1]
Radio Platform	R-FUNC-RFT-02, R-FUNC-RFT-03, R-FUNC-RA-01, R-FUNC-RA-03, R-FUNC-FB-04, R-FUNC-FB-05, R-FUNC-RFT-05, R-FUNC-RFT-06	The requirements are described in clauses 6.5.2 and 6.5.3 of ETSI EN 302 969 [1]
Radio Library	R-FUNC-FB-06	The requirement is described in clause 6.3.6 of ETSI EN 302 969 [1]
	R-FUNC-FB-01	The requirement is described in clause 6.3.1 of ETSI EN 302 969 [1]
Radio Virtual Machine	R-FUNC-MDR-13	The requirement is described in clause 6.4.13 of ETSI EN 302 969 [1]
Configcodes	R-FUNC-MDR-01, R-FUNC-MDR-02, R-FUNC-MDR-04	The requirements are described in clauses 6.4.1, 6.4.2 and 6.4.4 of ETSI EN 302 969 [1]
Radio Applications	R-FUNC-RA-02, R-FUNC-FB-03	The requirement is described in clause 6.2.2 of ETSI EN 302 969 [1]
Interfaces	R-FUNC-MDR-04	The requirement is described in clause 6.4.4 of ETSI EN 302 969 [1]
	R-FUNC-RFT-01	The requirement is described in clause 6.5.1 of ETSI EN 302 969 [1]

4.5 Reconfigurable Mobile Devices - the Radio Virtual Machine

4.5.1 Radio Virtual Machine basic principles

The RVM shall enable a RA to choose one among multiple available protection classes for code to be executed on the RVM as well as a protection class for the RF front-end. Depending on the combination of chosen RF & RVM protection classes, the required re-certification process of the software reconfigurable radio platform will be more or less complex. The basic principle is illustrated in Figure 4.5.

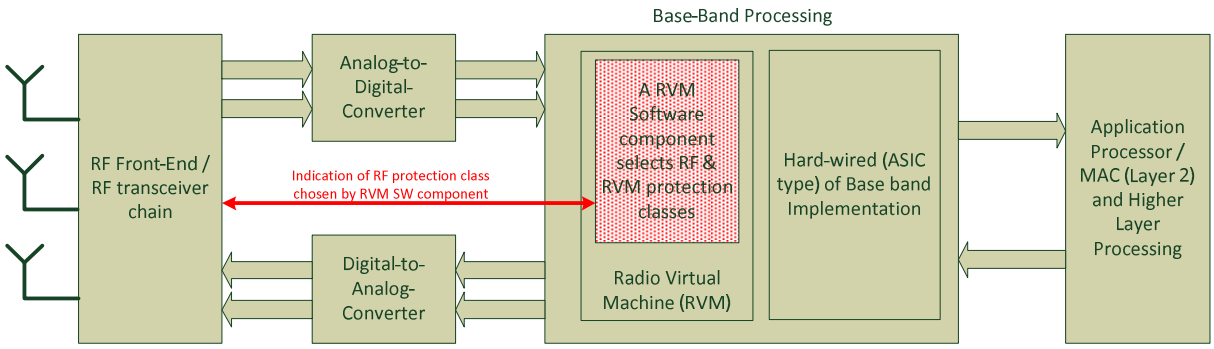

Figure 4.5: A typical radio equipment architecture comprising an RVM Software Component selecting RF and/or RVM protection class(es)

A typical radio equipment architecture includes an RF Transceiver chain, Analog-to-Digital converters, Digital-to-Analog converters, Base Band Processing, etc. An RVM controls RF Transceiver chain, in particular for selection of an RF Protection Class.

4.5.2 RVM System Requirement Mapping

The RVM above described is mapped to the system requirements described in ETSI EN 302 969 [1] as shown in Table 4.5.

Table 4.5: Mapping of RVM to the system requirements described in ETSI EN 302 969 [1]

Entity/Component/Unit	System Requirements [1]	Comments
Radio Virtual Machine	R-FUNC-MDR-13, R-FUNC-MDR-14	The requirement is described in clause 6.4.13 of ETSI EN 302 969 [1]
	R-FUNC-MDR-13, R-FUNC-MDR-15, R-FUNC-RFT-09	The requirements are described in clauses 6.4.13, 6.4.15 and 6.5.9 of ETSI EN 302 969 [1]

4.6 Reconfigurable Mobile Devices - Unified Radio Applications

4.6.1 Introduction

As already described in clause 4.3.2, RAs loaded into a Reconfigurable MD are called URAs.

The procedure of distributing and executing RA codes consists of 3 steps: design time, installation time, and run time. Figures 4.6, 4.7, 4.8, 4.9 and 4.10 illustrate these three steps for the case of platform-specific executable code, platform-independent source code (static / dynamic linking), and platform-independent IR (static / dynamic linking), respectively.

4.6.2 Distribution and Installation of RAP

In this clause, the procedure of distribution and installation of RA codes on the target reconfigurable MDs is presented. During the design time, the RA codes provider will generate a Radio Application Package (RAP) that includes metadata (e.g. for pipeline configuration) and RA codes. Note that the RC codes are part of the RA codes. In case that RC codes are executed in the non-real-time environment, they are compiled to be executed in a given AP before they are included in the RAP.

During the installation time, the RAP will be downloaded from a RadioApp Store and installed in the reconfigurable MD. The RA codes, including RC codes, and metadata (e.g. for pipeline configuration) included in the RAP are installed in the reconfigurable MD. Note that the RC codes are installed in the AP for operations that do not have to be executed in real time processing such as context information processing, while the Functional Block (SFBs & UDFBs) codes shall be installed in the Radio Computer to be processed in real-time.

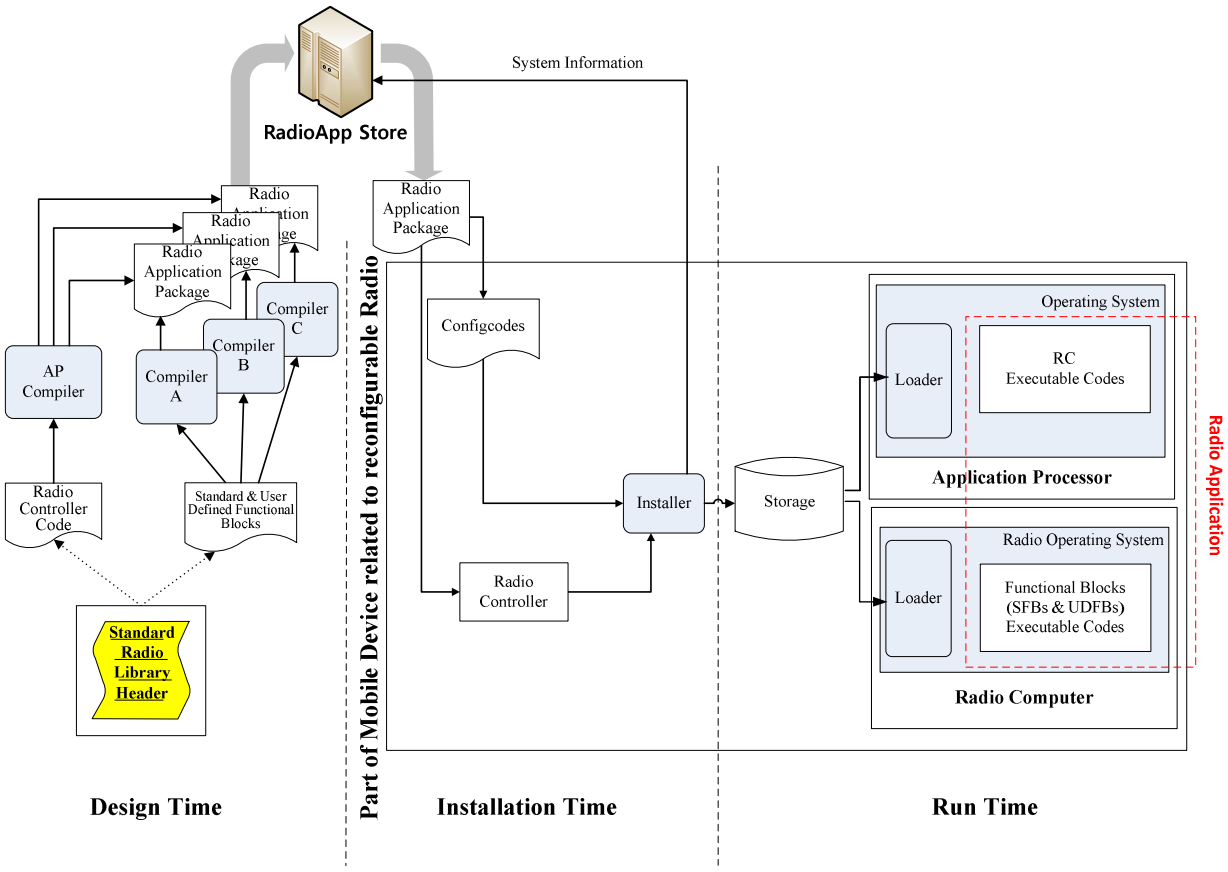

Figure 4.6: Conceptual diagram for adopting platform-specific executable code for radio application package for MDRC-2, MDRC-3 and MDRC-4

Figure 4.6 illustrates a block-diagram corresponding to the case of distributing Configcodes that are executable in a given reconfigurable MD. When the Configcodes are executable, the Functional Blocks (SFBs & UDFBs) are executed on the Radio Computer. They are compiled for each target platform during the design time to generate the corresponding Configcodes. This means that UDFB and SFB code are compiled in accordance with a given Radio Computer before they are included in the RAP during the design time. After compilation, the Configcodes including both UDFB and SFB codes are installed and loaded into reconfigurable MD to be operated on the ROS.

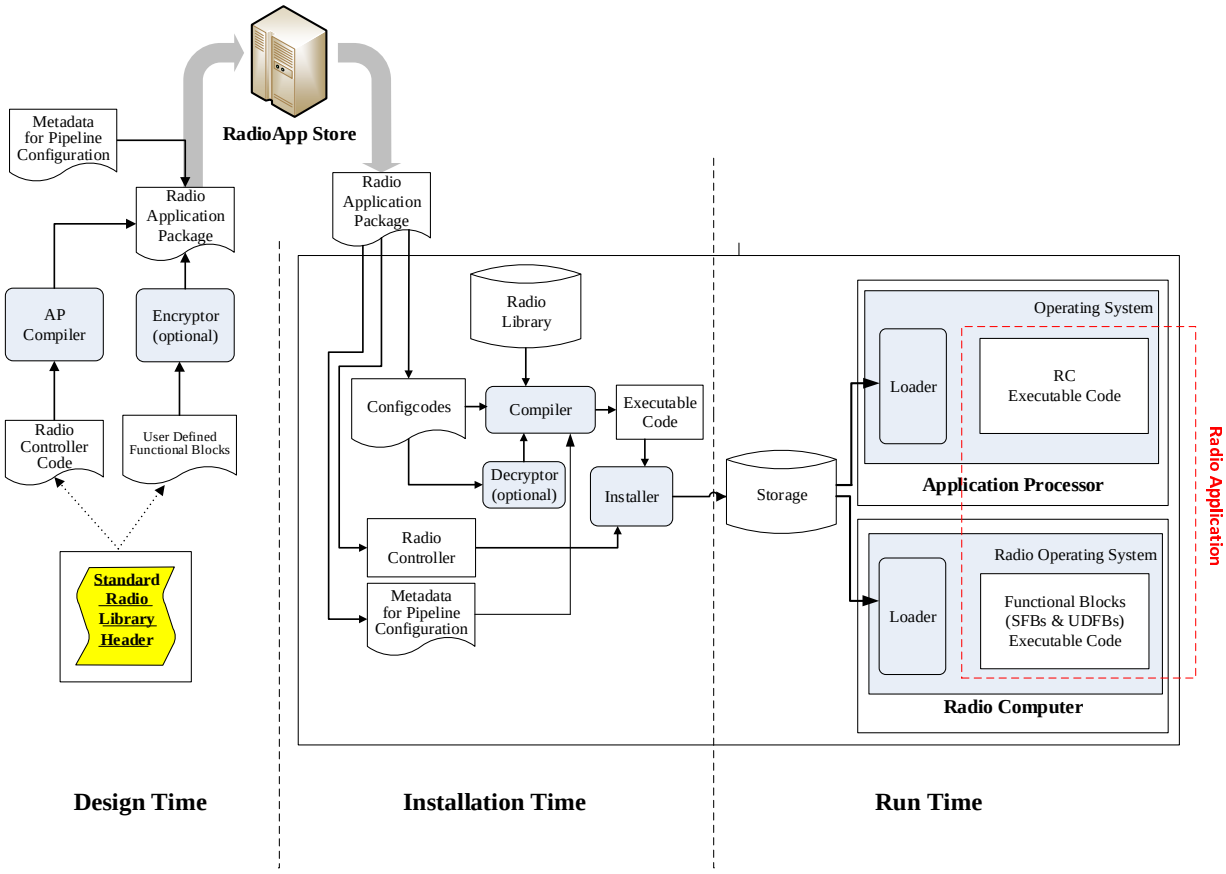

Figure 4.7: Conceptual Diagram of adopting platform-independent source code (static linking) for radio application package for MDRC-5, MDRC-6 and MDRC-7

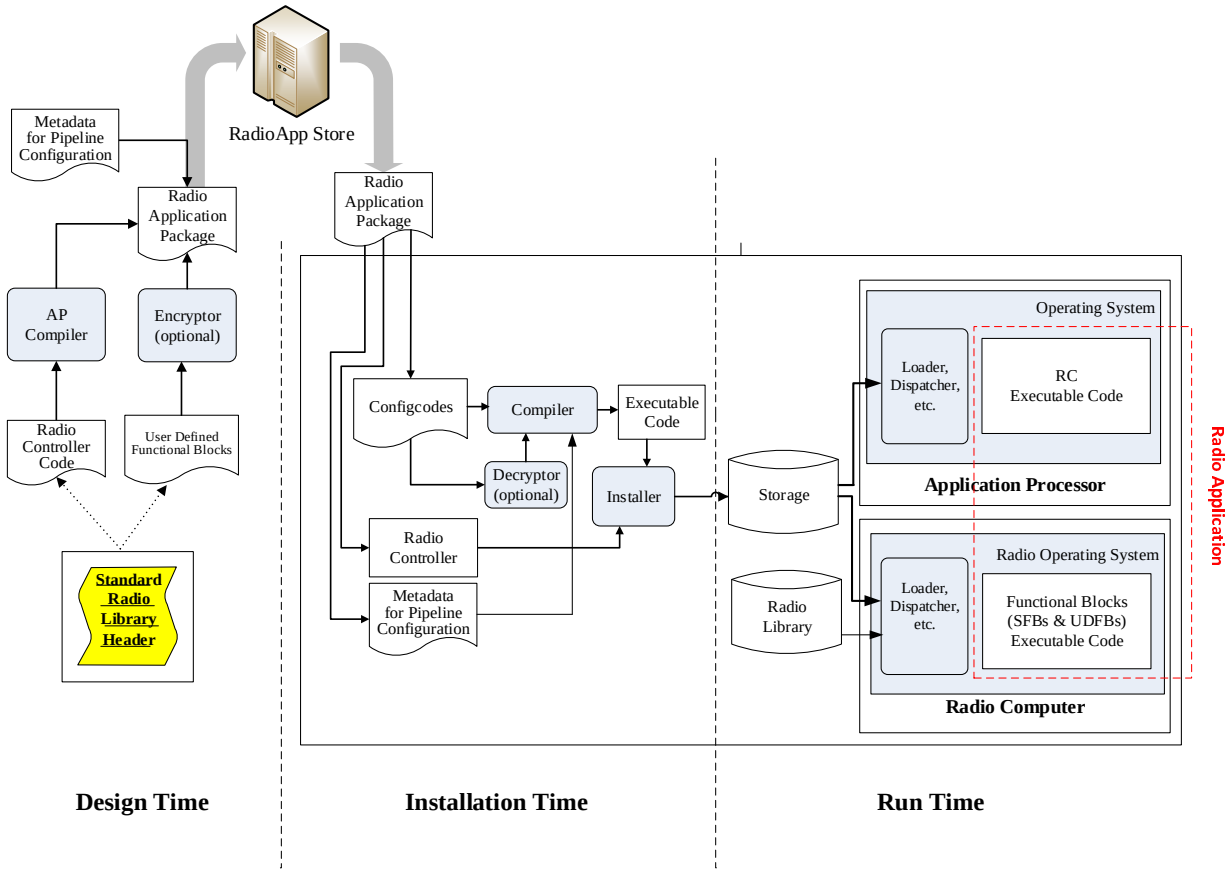

Figure 4.8: Conceptual Diagram of adopting platform-independent source code (dynamic linking) for radio application package for MDRC-5, MDRC-6 and MDRC-7

Figure 4.7 and Figure 4.8 illustrate a block-diagram corresponding to the case of distributing Configcodes in a form of platform-independent source code for static and dynamic linking respectively. When the Configcodes are provided in a platform-independent source code, the RA codes include the RC and UDFB codes only. As for the SFBs, the metadata provides information for efficient compilation. The function calls of the SFBs that are needed to execute the target URA are contained in the Configcodes. The Configcodes consisting of the UDFBs are compiled (e.g. in reconfigurable MD or in the cloud) during the installation time. The native implementation of SFBs is done before run time and is contained in the native library.

In the case of static linking, as illustrated in Figure 4.7, the linking of UDFBs with SFBs is performed during installation time. During the run time, the compiled codes are loaded to be executed on the ROS.

In the case of dynamic linking, as illustrated in Figure 4.8, the linking of UDFBs with SFBs is performed during run time.

In both cases, the compilation process during Installation time can be done using one of the two procedures as discussed in clause 4.4 which is summarized as follows:

1) Installation Time functions is performed within the Mobile Device or alternatively.

2) Some of the Installation Time functions except for the function of Installer itself are performed externally (i.e. in the "Cloud"). Typically, such a Cloud service is controlled by the platform vendor.

Depending on the upper choice, the Mobile Device Architecture may change. For example, with compilation being executed in the Cloud, no compiler is required in the Mobile Device. Note that, for the cloud service of compiling URA configcodes in the case of dynamic linking, Radio Library Native Implementation should be provided within MD.

As shown in Figures 4.7 and 4.8, the RA code might be optionally encrypted. If the RA code was originally encrypted then the corresponding Configcodes should be decrypted before the compilation during the installation time.

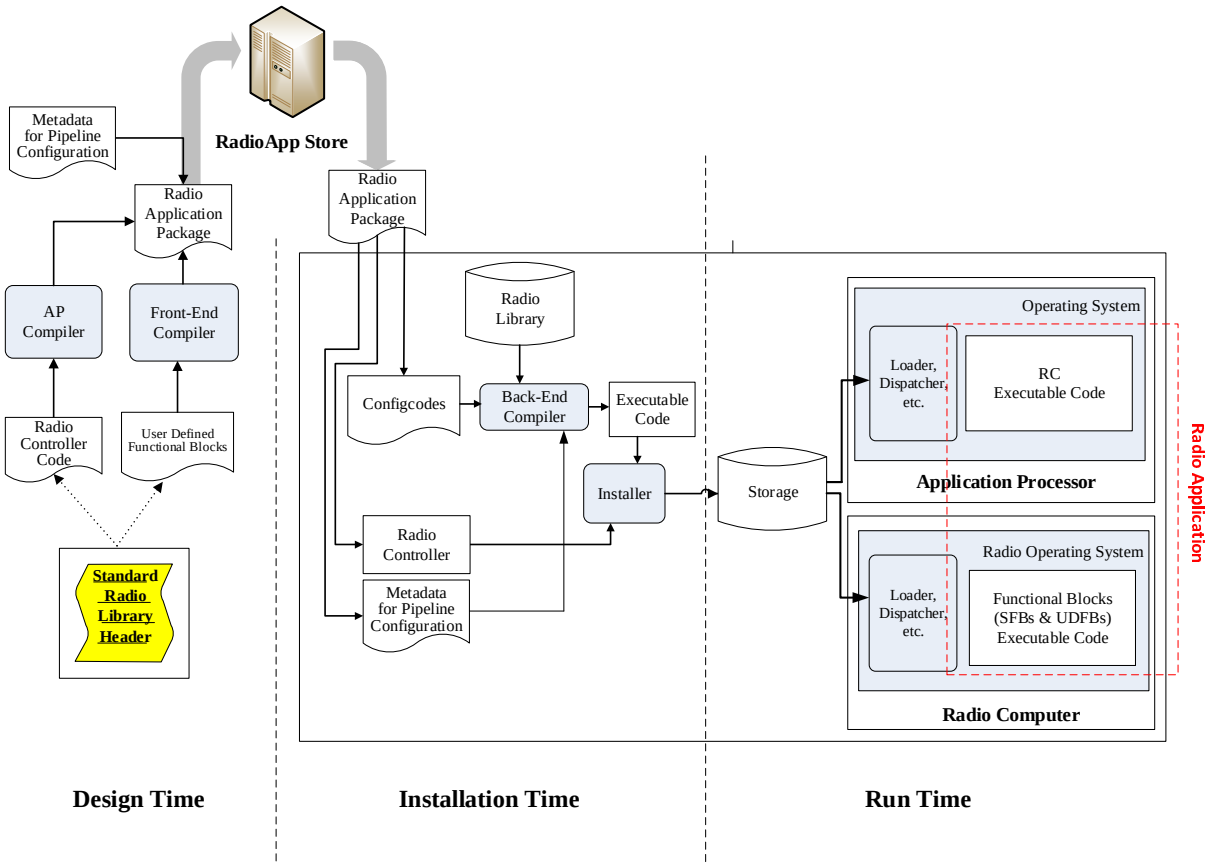

Figure 4.9: Conceptual Diagram of adopting platform-independent IR (static linking) for radio application package for MDRC-5, MDRC-6 and MDRC-7

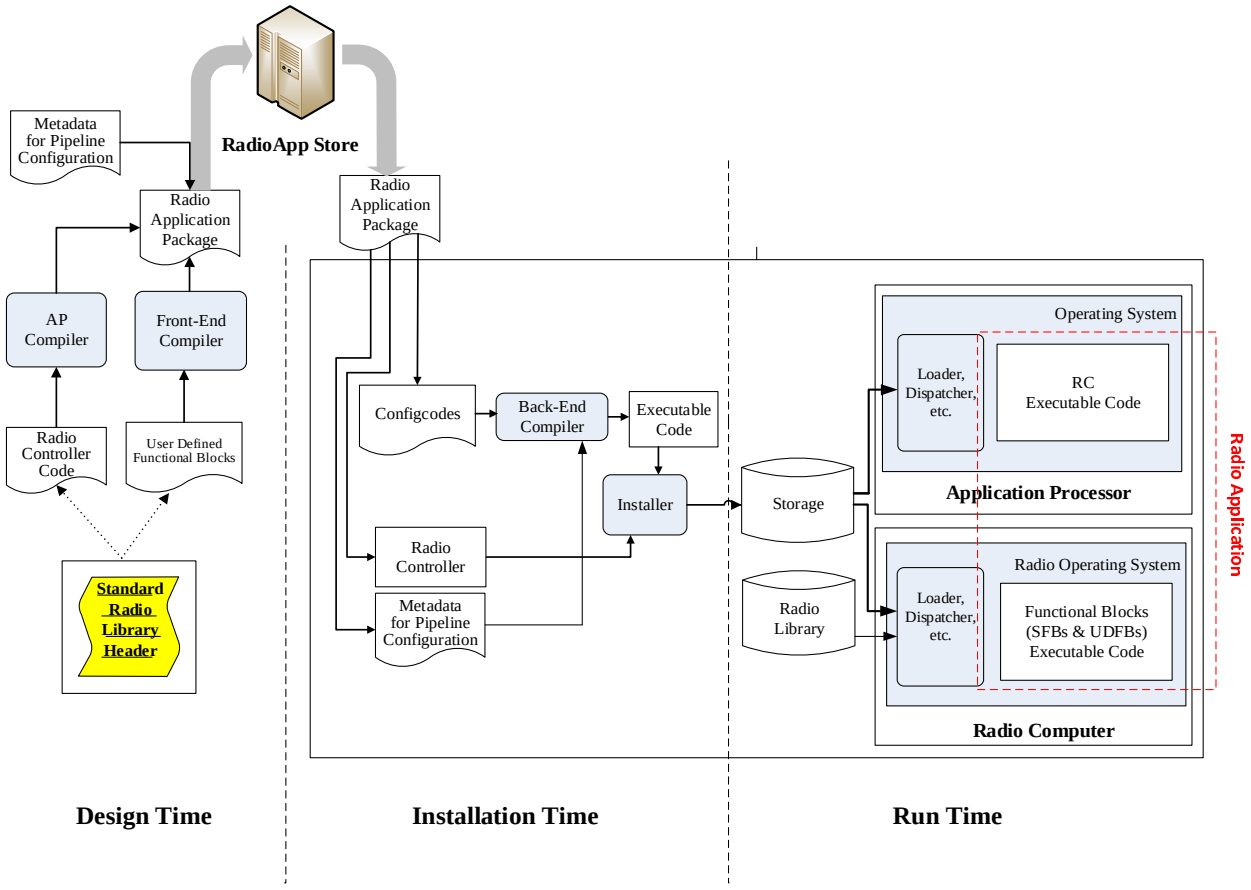

Figure 4.10: Conceptual Diagram of adopting platform-independent IR (dynamic linking) for radio application package for MDRC-5, MDRC-6 and MDRC-7

Figure 4.9 and Figure 4.10 illustrate a block-diagram corresponding to the case of distributing Configcodes in a form of platform-independent IR for static and dynamic linking respectively. When the Configcodes are provided in the platform-independent IR, the RA codes which include the UDFB codes are front-end compiled during the design time. At the installation time, the front-end compiled UDFB codes of Configcodes are back-end compiled at reconfigurable MD to be translated into an executable code specific to a given Radio Computer. The native implementation of SFBs is done before run time and is contained in the native library.

In the case of static linking, as illustrated in Figure 4.9, the linking of UDFBs with SFBs is performed during installation time. During the run time, the back-end compiled codes are loaded to be executed on the ROS.

In the case of dynamic linking, as illustrated in Figure 4.10, the linking of UDFBs with SFBs is performed during run time.

In both cases, the compilation process during Installation time can be done using one of the two procedures as discussed in clause 4.4 which is summarized as follows:

1) Installation Time functions is performed within the Mobile Device or alternatively.

2) Some of the Installation Time functions except for the function of Installer itself are performed externally (i.e. in the "Cloud"). Typically, such a Cloud service is controlled by the platform vendor.

Depending on the upper choice, the Mobile Device Architecture may change. For example, with compilation being executed in the Cloud, no compiler is required in the Mobile Device. Note that, for the cloud service of compiling URA configcodes in the case of dynamic linking, Radio Library Native Implementation should be provided within MD.

In the case of adopting platform-independent IR, the UDFB codes of Configcodes are back-end compiled for a given Radio Computer during the installation time.

4.6.3 Operational Structure of URA

In this clause the operational structure of URA in run time is presented. Two different cases are considered:

1) The URA configcodes are executable on a given MD.

2) The URA configcodes are source codes or Intermediate Representation (IR) that is to be compiled at a given MD.

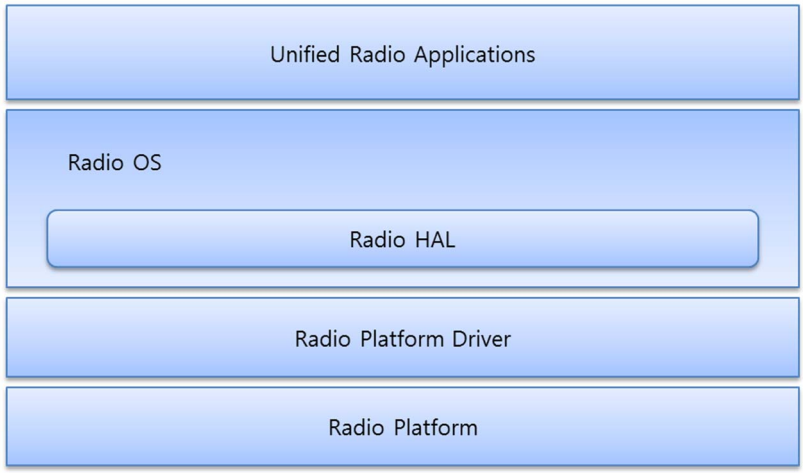

Figure 4.11: Operational structure of URA when URA configcodes are executable on a target platform

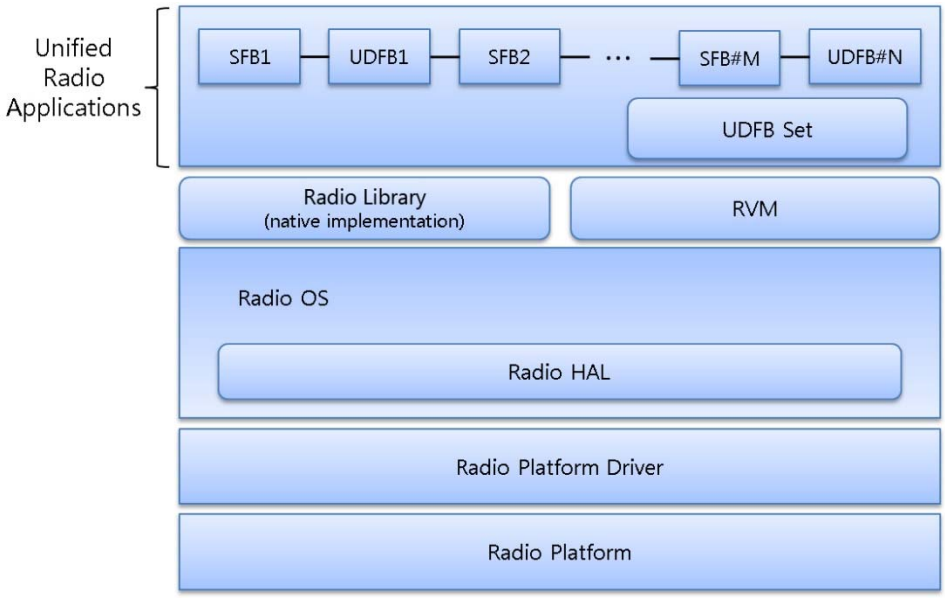

Figure 4.12: Operational structure of URA when URA configcodes are source codes or IR to be compiled

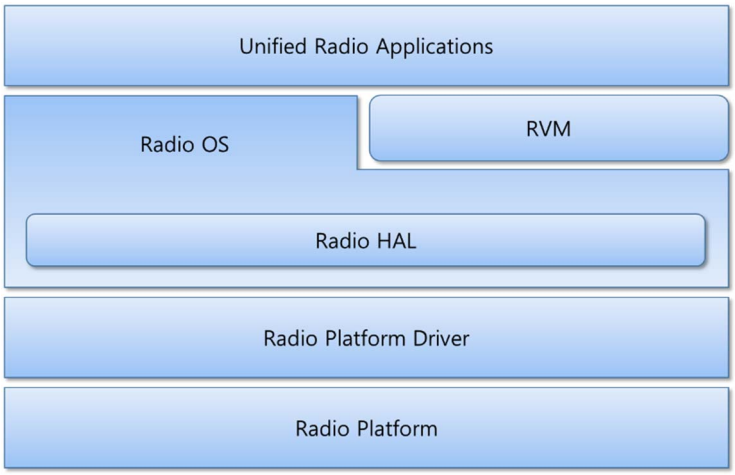

Figure 4.13: Operational structure of URA when URA configcodes are combined (executable & IR) codes

Note that SFBs are classified into two groups, i.e. requiring or not dedicated hardware accelerators. In case that a hardware accelerator is used, it is accessed through the Radio Hardware Abstraction Layer (HAL). In the other case, platform specific code is provided for the concerned SFB by the Radio Library.

The first case (i.e. executable code is provided) is illustrated in Figure 4.11. Here the SFBs and UDFBs needed to perform a given URA are already bound in the executable configcodes of URA.

The second case (i.e. Source Code or IR is provided) is illustrated in Figure 4.12. In this case, the UDFBs needed to perform a given URA are included in the configcodes of the URA and shall be compiled (see also Figure 4.3) for Source Code (by the Compiler) or IR (by the Back End Compiler) respectively. Note that the native implementation of Radio Library shall be prepared in a given MD separately because the Radio Library native implementation cannot be contained in URA configcodes. As mentioned earlier, the function calls of SFB(s) are provided in the metadata. Generally, the native implementation of Radio Library is provided by the Radio Computer vendor because Radio Library includes SFB(s) that is/are implemented on the Radio Computer. These SFBs can be implemented without using hardware accelerator(s) or for combining accelerator(s) and program code to generate another SFB(s).

The third case is illustrated in Figure 4.13 which is a hybrid of the former two cases. Here, URA consists of executable codes and IR codes. Operational procedure for the executable codes in this case is equivalent to that of the first case, i.e. shown in Figure 4.11, whereas the IR part shall be processed in the RVM. In this way, IR can be executed in run-time. The RVM may be implemented, for example, as an interpreter, a just-in-time compiler, etc.

In the above explained 3 cases a Radio Hardware Abstraction Layer (HAL) includes hardware abstraction for SFBs implementation using hardware accelerator(s). This means that, whenever the SFB(s) to be implemented using hardware accelerator(s) is/are called in a given URA code, they are implemented directly on a corresponding hardware accelerator(s) via the Radio HAL. As it will be discussed later in this clause, the Radio HAL includes also a hardware abstraction for the UDFB(s) that is/are composed of a set of SFBs at least one of which is implemented using the hardware accelerator(s).

SFBs typically include all those functional blocks which are commonly used in URA (e.g. such as Fast Fourier Transform (FFT), etc.) and those functionalities that are implemented very efficiently using special-purpose accelerator(s) in a given radio platform (e.g. such as Turbo coder). SFBs can thus be implemented in Software or dedicated Hardware.

The UDFB Set shown in Figure 4.12 includes all the UDFBs to be used in a given URA. It is important that any SFB can be modified and/or extended as appropriate by replacing it with UDFB(s). Therefore, UDFB(s) could be good candidate(s) for SFBs extension, which means that they might become SFBs later (and become "atomic" as the normal SFBs) through an extension of the Standard Radio Library (e.g. by approval through the community). Since any UDFB Set is to be provided by the RA provider (e.g. a 3rd party different from the Radio Computer vendor), in order for RCF to be able to perform basic controls of every UDFB, control interface functions such as *"start"*, *"stop"*, *"pause"*, *"get_port"* and *"initialize"* may have to be specified for the corresponding UDFB(s).

The Operational structure of URA depicted in Figure 4.12 includes the following components:

- **URA** include sSFB(s) and UDFB(s) in accordance with the contents of metadata in a given RAP.

- **Radio Library** contains a platform specific code of SFBs that will be implemented on the Radio Computer.

NOTE: Those SFBs which are implemented using hardware accelerator(s) are supported by the Radio HAL. In this case, the Radio Library typically contains corresponding function calls to access hardware accelerator(s).

- **UDFB Set** includes all the UDFBs to be used in given URA and is in general provided by the RA provider. UDFBs are included in RAP together with metadata and RC code. Since UDFBs are in general modified and/or extended version of SFBs, UDFBs in many cases have a dependency on SFBs.

- **Radio HAL** is to abstract radio platform and it shall support SFBs to be implemented using hardware accelerator in order for each of those SFBs to be implemented directly on corresponding hardware accelerator(s). The Radio HAL is platform specific and is not standardized.

- **Radio Platform Driver** is a hardware driver used by the ROS to access the radio platform.

- **Radio Platform** in general includes RF transceiver, antenna(s), fixed and/or configurable hardware accelerator and/or programmable IP core(s).

Figure 4.14 illustrates an implementation of functional blocks on a given Radio Computer. In the example shown in Figure 4.14, the number of SFBs for programmable components has been set to M_1 and the number of SFBs requiring dedicated hardware accelerators has been set to M_2, while the total number of SFBs is $M = M_1 + M_2$. As mentioned earlier in this clause, some SFBs, for example FFT, Turbo decoder, Multi-Input-Multi-Output (MIMO) decoder, etc., can be implemented directly on the corresponding hardware accelerator, for example to achieve high performance and low power consumption. Those SFBs that are executed by the hardware accelerator(s) are supported by the Radio HAL for the implementation on the corresponding dedicated accelerator(s). This means that, when each SFB to be implemented on the dedicated accelerator(s) is called in URA, it is implemented directly on the corresponding dedicated accelerator(s) through Radio HAL. Similarly, operations such as bit-reverse, multiply and accumulation, etc., are introduced by SFBs, e.g. for programmable components.

Consequently, the SFB/UDFB execution codes required on the Radio Computer consists of the following two parts: one part is execution codes implemented on programmable components and the other part is Radio HAL codes implemented on dedicated accelerators. It can be summarized as follows:

{C: execution codes required on Radio Computer for SFBs/UDFBs implementation} = {A: execution codes for SFBs/UDFBs for programmable components} + {B: Radio HAL codes for SFBs/UDFBs requiring dedicated hardware accelerators },

meaning that:

$$C = A + B$$

where the portion of A and B will be determined by each vendor. It particularly means that:

{SFBs/UDFBs} = {SFBs/UDFBs for programmable components} ∪ {SFBs/UDFBs requiring dedicated hardware accelerators }

where:

{SFBs/UDFBs for programmable components } ∩ {SFBs/UDFBs requiring dedicated hardware accelerators } = ∅.

The reason why SFBs are classified into two groups, (i.e. requiring or not dedicated hardware accelerators), is that each category has its own pros and cons. The former, since it is implemented on dedicated hardware accelerators, is advantageous, e.g. for power consumption, speed-up operation, and, cost-effectiveness while the latter is advantageous mainly for flexibility. It is expected that the dedicated hardware accelerator(s) will be used relatively more widely at the beginning stage until programmable devices become competitive to dedicated hardware devices in performance. As semiconductor technology evolves, the SFBs for programmable components will gradually become more and more dominant in a long term standpoint.

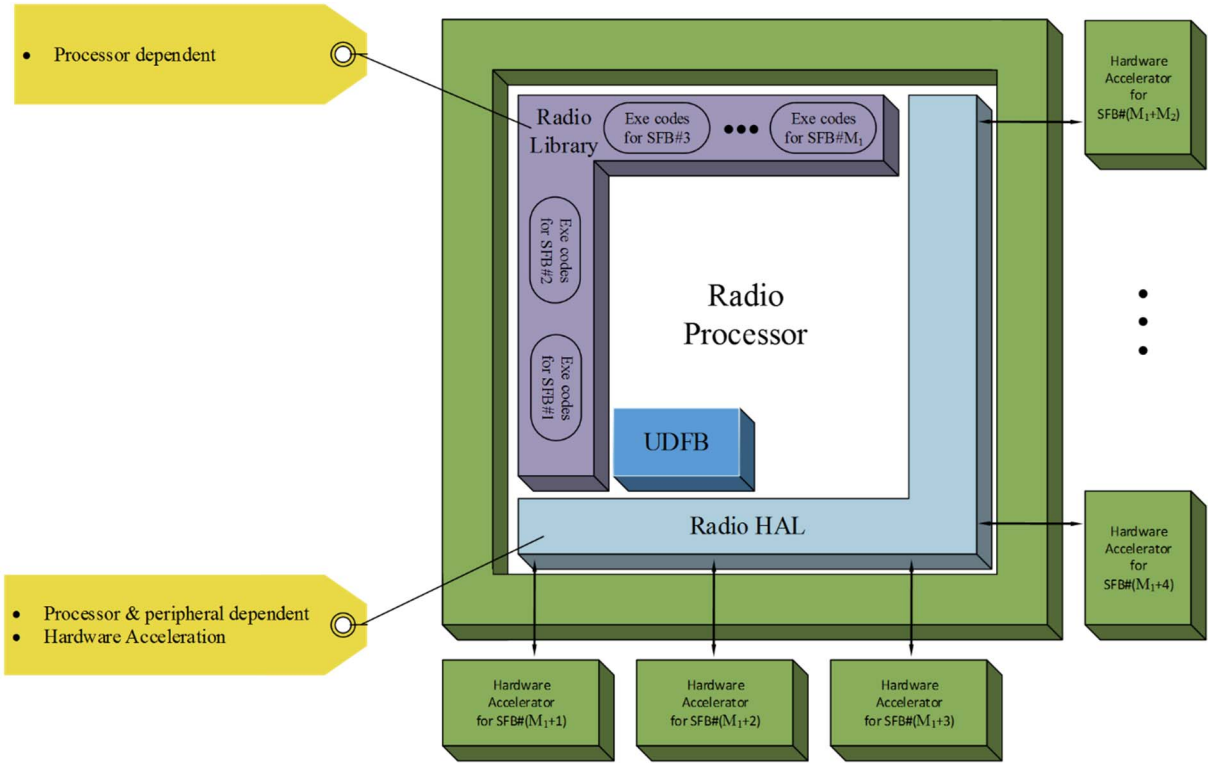

Figure 4.14: Implementation of functional blocks libraries on Radio Computer

4.6.4 URA System Requirement Mapping

The URA above described is mapped to the system requirements described in ETSI EN 302 969 [1] as shown in Table 4.6.

Table 4.6: Mapping of URA to the system requirements described in ETSI EN 302 969 [1]

Entity/Component/Unit	System Requirements [1]	Comments
Configcodes	R-FUNC-MDR-01 R-FUNC-MDR-02 R-FUNC-MDR-04	The requirements are described in clauses 6.4.1, 6.4.2 and 6.4.4 of ETSI EN 302 969 [1].
	R-FUNC-RA-02	The requirement is described in clause 6.2.2 of ETSI EN 302 969 [1].
Functional Blocks	R-FUNC-MDR-13 R-FUNC-MDR-12	The requirement is described in clause 6.4.13 of ETSI EN 302 969 [1].
	R-FUNC-FB-02	The requirement is described in clause 6.3.2 of ETSI EN 302 969 [1].
Radio Library	R-FUNC-FB-06	The requirement is described in clause 6.3.6 of ETSI EN 302 969 [1].
	R-FUNC-FB-01	The requirement is described in clause 6.3.1 of ETSI EN 302 969 [1].
Radio Applications	R-FUNC-MDR-07	The requirement is described in clause 6.4.7 of ETSI EN 302 969 [1].
	R-FUNC-RA-06 R-FUNC-RA-05 R-FUNC-RA-02	The requirements are described in clauses 6.2.6, 6.2.5 and 6.2.2 of ETSI EN 302 969 [1].

4.7 Security architecture for reconfigurable mobile devices

4.7.1 Description

The security architecture for reconfigurable mobile devices extends the architecture defined in clause 4.2 of the present document in order to ensure the security of the reconfigurable platform and of the digital assets.

NOTE: The rationale for the selection of security component, and the description of the security functions' operations, are detailed in ETSI TR 103 087 [i.5] with their normative counterpart being specified in ETSI TS 103 436 [i.4].

The security architecture relies on the following additional entities on the device:

- The Administrator Security Function (ASF), as sub-entity of the Administrator on the device end-point responsible for ensuring confidentiality, integrity, and authenticity of assets such as the RE Configuration Policy, and RAP(s), and supporting the non-repudiation, remote attestation, and configuration enforcement strategies.
- The RRS Configuration Manager (RRS-CM), a sub-entity of the Administrator in charge of long-term management.
- The Root of Trust, providing security services, such as secure storage, with a high level of security assurance.

Figure 4.15 illustrates the additional entities of the security architecture on the device.

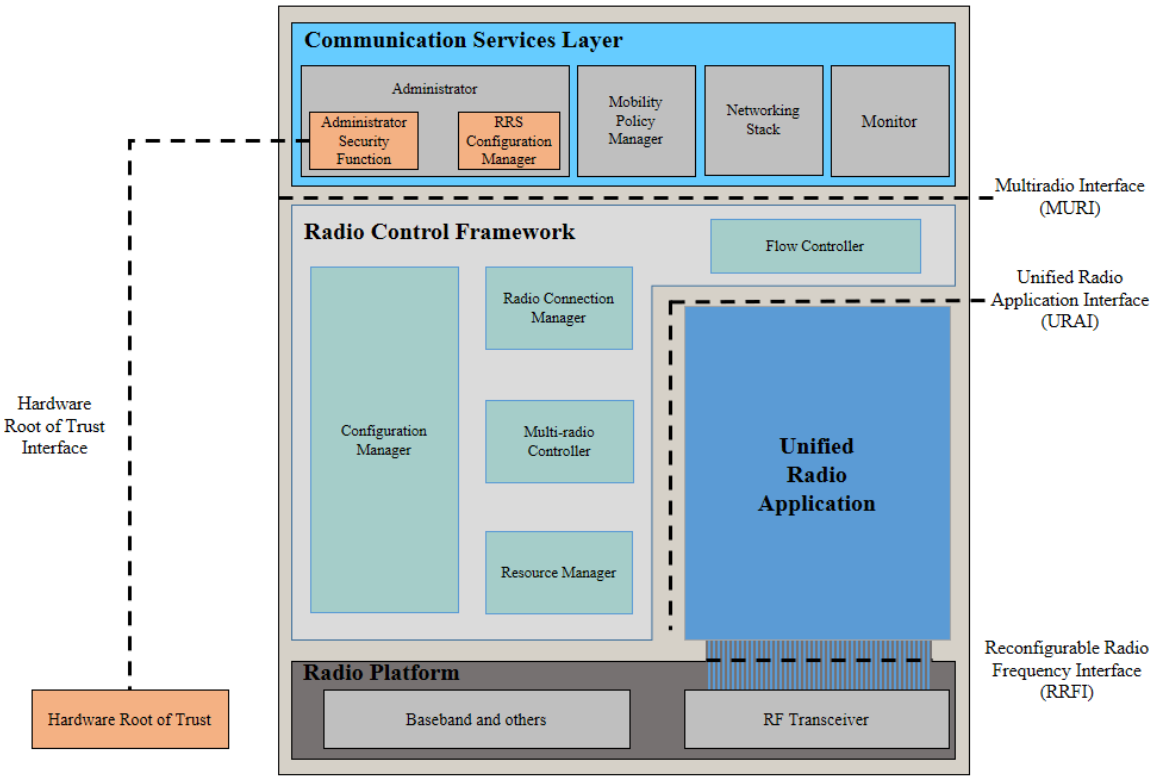

Figure 4.15: Reconfigurable Mobile Device Architecture with security components

The Hardware Root of Trust Interface is not covered in the present document.

Furthermore, the security architecture relies on the following external entities [i.4] and [i.5]:

- Asset Endorsement Functions, responsible for the security of digital assets (such as RE Configuration Policy, RAP and other assets).
- Command Security Functions, responsible for the security of remote commands being sent to the device.
- Verifying Entity and Requestor, supporting remote attestation.
- Manufacturer Key Management System (OEM KMS), RRS Configuration Authority (RRS-CA), and RRS Configuration Provider (RRS-CP), supporting the long-term management framework.

4.7.2 Security Components System Requirements mapping

The logical entities above described are mapped to the system requirements described in clause 6.6 of ETSI EN 302 969 [1] as shown in Table 4.7.

Table 4.7: Mapping of Security Components to the system requirements described in ETSI EN 302 969 [1]

Entity/Component/Unit	System Requirements [1]	Comments
Administrator Security Function (ASF)	R-FUNC-SEC-01 R-FUNC-SEC-02 R-FUNC-SEC-03 R-FUNC-SEC-04	In addition to supporting operations related to asset protection, the ASF acts as a proxy to other security functions on the platform, as detailed in [i.5]. The requirements are described in clauses 6.6.1, 6.6.2, 6.6.3 and 6.6.4 of ETSI EN 302 969 [1].
RRS Configuration Manager (RRS-CM)	R-FUNC-SEC-03	The operations of long-term management are detailed in [i.5], clause 11, with related requirements in [i.4]. The requirement is described in clause 6.6.3 of ETSI EN 302 969 [1].
Root of Trust	R-FUNC-SEC-04	The exact nature of the Root of Trust depends on industry best practices and the concept of security tiers introduced in [i.4]. The requirement is described in clause 6.6.4 of ETSI EN 302 969 [1].
Asset Endorsement Functions	R-FUNC-SEC-01 R-FUNC-SEC-02	These functions are the ASF's external peers. In particular, they provide proof that assets originate from authorized sources. The requirements are described in clauses 6.6.1 and 6.6.2 of ETSI EN 302 969 [1].
Command Security Functions	R-FUNC-SEC-03	Commands are processed by the Administrator after having being validated by the ASF. Command Security Functions provide, in particular, proof that configuration commands originate from authorized sources. The configuration enforcement framework is detailed in [i.5], clause 10, with related requirements in [i.4]. The requirement is described in clause 6.6.3 of ETSI EN 302 969 [1].
Verifying Entity Requestor	R-FUNC-SEC-03	The requestors typically wish to assert high level claims about the platform. The Verifying Entity translates these claims into atomic attestation requests towards platform components. Remote attestation is detailed in [i.5], clause 9, with related requirements in [i.4]. The requirement is described in clause 6.6.3 of ETSI EN 302 969 [1].
Manufacturer Key Management System (OEM KMS) RRS Configuration Authority (RRS-CA) RRS Configuration Provider (RRS-CP)	R-FUNC-SEC-03	The operations of the long-term management framework are detailed in [i.5], clause 11, with related requirements in [i.4]. The requirement is described in clause 6.6.3 of ETSI EN 302 969 [1].

5 Reference Points

5.1 Introduction

Figure 5.1 illustrates the entire architecture of MD with all the reference points being specified between corresponding entities. Each solid line between two blocks denotes a reference point (i.e. a logical or physical interface) defined between the two blocks through which direct interaction(s) between the two blocks is(are) performed, whereas each dotted line between two blocks denotes that interaction(s) between the two blocks is performed through ROS based on (a) command(s) issued by a corresponding block. As it will be shown, blocks in RCF, i.e. CM, RCM, MRC, and RM, issue the command for the interaction(s) to take place at URA through ROS. The definition of each reference point is based on the three kinds of interfaces, i.e. MURI which are interfaces between entities of CSL and that of RCF, URAI which are interfaces between URA and entities of RCF, and Reconfigurable Radio Frequency Interfaces (RRFI) which are interfaces between URA and Radio Frequency (RF) part. In addition to MURI, URAI, and RRFI, interfaces between entities of RCF have also been defined as reference points. In the present document, the reference points are classified according to procedures of their functions such that the classification of each of the reference points becomes coincident with each of the procedures defined in clause 6.

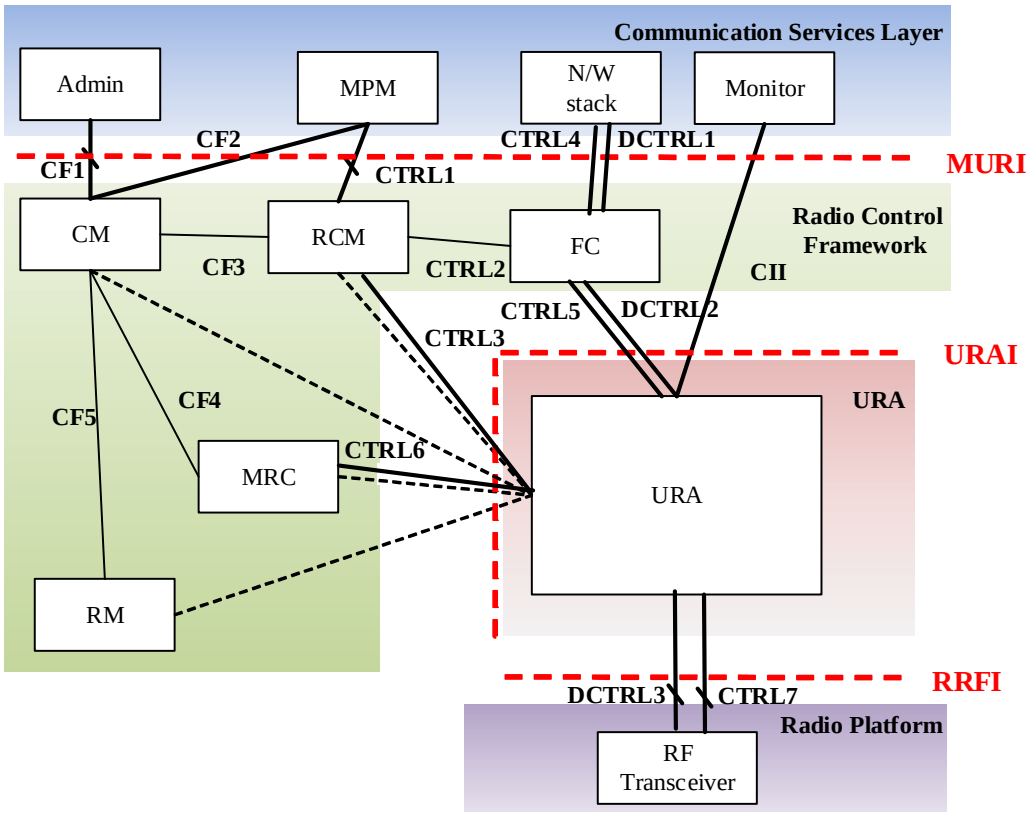

Figure 5.1: Entire architecture of reference points for the MD

5.2 Reference Points required for Installation/uninstallation and creating/deleting an instance of a URA

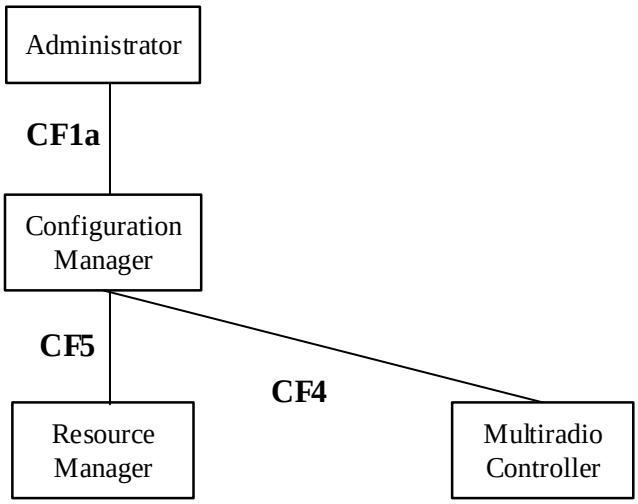

Figure 5.2: Illustration of reference points for installation/uninstallation and creating/deleting instance of a URA

Figure 5.2 illustrates reference points, CF1a, CF1b, CF4, and CF5, which are related to the installation/uninstallation (CF1a) or creation/deletion (CF1b, CF4, CF5) instance of a URA.

Reference Point **CF1a** is an interface between Administrator and CM, through which Administrator requests CM to perform the installation/uninstallation of a URA and receives a response from CM.

Reference Point **CF1b** is an interface between Administrator and CM, through which Administrator requests CM to create/delete an instance of an URA and receives a response from CM.

Reference Point **CF4** provides interaction between CM and MRC, through which CM requests MRC to send the parameters related to radio resources to CM, and receives a response (i.e. the requested parameters) from MRC during the procedure of creating an instance of URA.

Reference Point **CF5** provides interaction between CM and RM, through which CM requests RM to send the parameters related to computational resources to CM, and receives a response (i.e. the requested parameters) from RM during the procedure of creating an instance of URA.

5.3 Reference Points required for list checking of URA

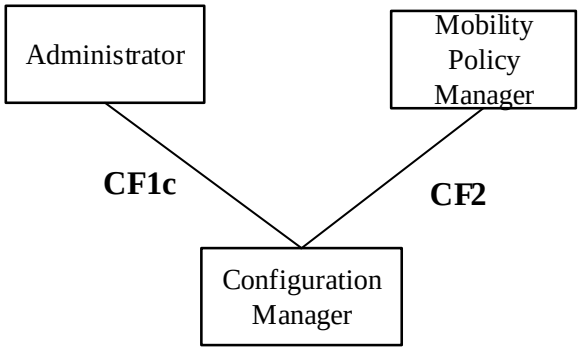

Figure 5.3: Illustration of reference points for obtaining the lists of URA

Figure 5.3 illustrates the reference points CF1c and CF2, which are related to URA list checking.

Reference Point **CF1c** is an interface between Administrator and CM, through which Administrator requests CM to send the URA list, and receives a response (i.e. the URA list), from CM.

Reference Point **CF2** is an interface between MPM and CM, through which MPM requests CM to send the URA list and receives a response (i.e. the URA list), from CM.

5.4 Reference Points required for activation/deactivation of URA

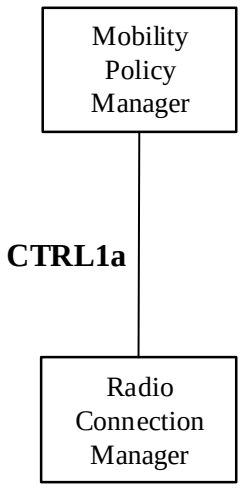

Figure 5.4: Illustration of reference point for activation/deactivation of URA

Figure 5.4 illustrates reference point CTRL1a, which is related to the activation/deactivation of URA.

Reference Point **CTRL1a** is an interface between MPM and RCM, through which MPM requests RCM to perform the activation/deactivation of a URA, and receives a response from RCM.

5.5 Reference Points required for transferring context information

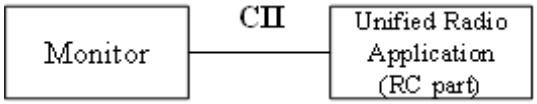

Figure 5.5: Illustration of reference points for context information transfer

Figure 5.5 illustrates reference point CII, which is related to the transfer of context information.

Reference Point **CII** is an interface between Monitor and RC in the URA, through which Monitor requests RC in URA to send context information and receives a response (i.e. the context information), from RC in the URA.

Explanation: The context information is generated from corresponding functional block(s) of URA and transferred to RC which, in turn, transfer it to Monitor upon request.

Note that CII is an interface between Monitor in the CSL and RC part in the URA as shown in Figure 5.5. While functional blocks in the Radio Application code is loaded as the URA for real-time processing, the RC part in the Radio Application code is loaded at the same level where the Monitor is located because the interaction between RC and Monitor is executed in non-real-time. Consequently, CII does not break the interface hierarchy defined in the present document.

5.6 Reference Points required for creating data flow and sending/receiving user data

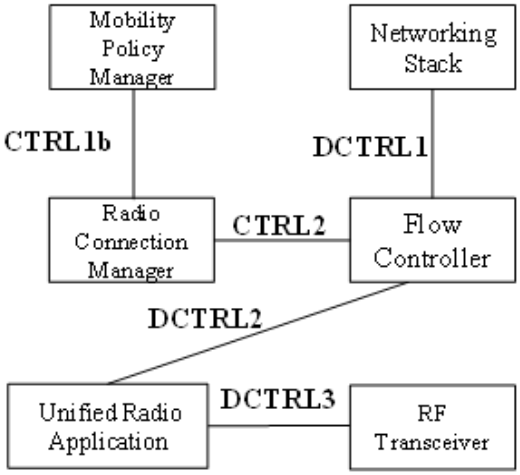

Figure 5.6: Illustration of reference points for creating data flow and sending/receiving user data

Figure 5.6 illustrates reference points CTRL1b, CTRL2, DCTRL1, DCTRL2, and DCTRL3, which are used for creating data flow and for sending and receiving user data as follows:

- Reference Point **CTRL1b** is an interface between MPM and RCM, through which MPM requests RCM to create a data flow or a network association with a peer equipment and receives a response from RCM.
- Reference Point **CTRL2** provides interaction between RCM and FC, which is used together with CTRL1b for creating a data flow.
- Reference Point **DCTRL1** is an interface between FC and Networking Stack, through which FC receives/transfers user data from/to Networking stack for the procedure of sending/receiving data. When Networking Stack sends data, the FC can optionally send back an acknowledgement at the end of transmission.
- Reference Point **DCTRL2** is an interface between FC and URA, through which FC transfers user data to URA and requests URA to transfer information related to user data (such as throughput, data bandwidth, etc.) to FC. Through the same interface the FC can also receive data from URA, i.e. user data (data reception) or user data information upon request (and before actual data transmission).
- Reference Point **DCTRL3** is an interface between URA and RF transceiver, through which URA receives/transmits user data from/to RF transceiver.

5.7 Reference Points required for radio environment measurements

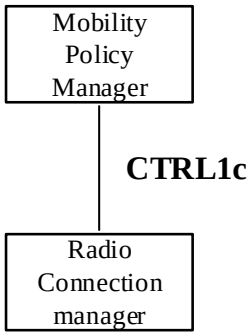

Figure 5.7: Illustration of reference points for radio environment measurements

Figure 5.7 illustrates reference point CTRL1c, which is related to the radio environment measurements.

Reference Point **CTRL1c** is an interface between MPM and RCM, through which MPM requests RCM to start/stop measurements for radio environment and receives a response from RCM.

5.8 Reference Points required for reporting discovered peer equipment

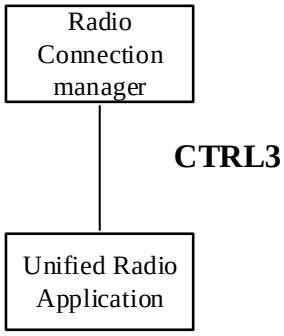

Figure 5.8: Illustration of reference points for reporting discovered peer equipments

Figure 5.8 illustrates reference point CTRL3, which is related to reporting discovered peer equipment.

Reference Point **CTRL3** is an interface between RCM and URA, through which RCM requests URA to report discovered peer equipment and receives a response (i.e. Information about discovered peer equipment) from URA.

5.9 Reference Points required for flexible data flow

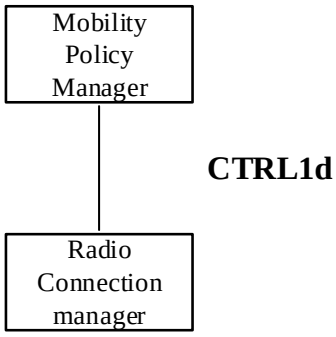

Figure 5.9: Illustration of reference points for flexible data flow

Figure 5.9 illustrates reference point CTRL1d, which is related to the flexible data flow.

Reference Point **CTRL1d** is an interface between MPM and RCM, through which MPM requests RCM to move/partition/combine data flow and receives a response from RCM.

5.10 Reference Points required for data flow control

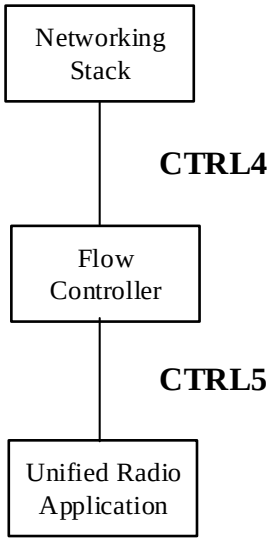

Figure 5.10: Illustration of reference points for data flow control

Figure 5.10 illustrates reference point CTRL4 and CTRL5, which are related to the data flow control.

Reference Point **CTRL4** is an interface between FC and N/W stack, through which FC requests N/W stack to change the configuration of data flow and receives a response from N/W stack.Reference Point **CTRL5** is an interface between URA and FC, through which URA requests FC to change the configuration of data flow and receives a response from FC.

5.11 Reference Points required for synchronizing radio time

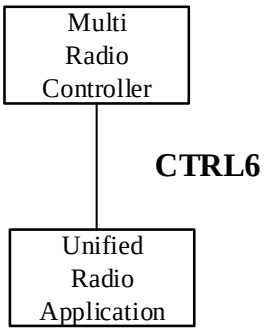

Figure 5.11: Illustration of reference points for synchronizing radio time

Figure 5.11 illustrates reference point CTRL6, which is related to the synchronizing radio time.

Reference Point **CTRL6** is an interface between MRC and URA, through which MRC requests URA to get synchronizing radio time and receives a response (i.e. synchronized radio time) from URA.

5.12 Reference Points required for control of reconfigurable RF transceiver

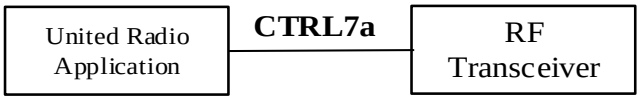

Figure 5.12: Illustration of reference points for spectrum control services

Figure 5.12 illustrates reference point CTRL7a, which is related to the spectrum control services.

Reference Point **CTRL7a** is an interface between URA and RF transceiver, through which URA requests RF transceiver to set up centre frequency/bandwidth/sampling rate and receives a response from RF transceiver.

Reference Point **CTRL7a** is an interface between URA and RF transceiver, through which URA requests RF transceiver to get Tx/Rx chain parameters and receives a response (i.e. the Tx/Rx chain parameters) from RF transceiver.

Figure 5.13: Illustration of reference points for power control services

Figure 5.13 illustrates reference point CTRL7b, which is related to the power control services.

Reference Point **CTRL7b** is an interface between URA and RF transceiver, through which URA requests RF transceiver to set up maximum transmit power/antenna power/rx gain and receives a response from RF transceiver.

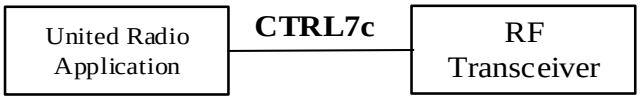

Figure 5.14: Illustration of reference points for antenna management services

Figure 5.14 illustrates reference point CTRL7c, which is related to the antenna management services.

Reference Point **CTRL7c** is an interface between URA and RF transceiver, through which URA requests RF transceiver to select tx/rx antenna port and receives a response from RF transceiver.

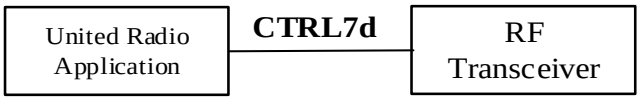

Figure 5.15: Illustration of reference points for chain control services

Figure 5.15 illustrates reference point CTRL7d, which is related to the chain control services.

Reference Point **CTRL7d** is an interface between URA and RF transceiver, through which URA requests RF transceiver to set start&stop time for transmission/reception, and receives a response from RF transceiver.

Reference Point **CTRL7d** is an interface between URA and RF transceiver, through which URA requests RF transceiver to update chain parameters and receives a response from RF transceiver.

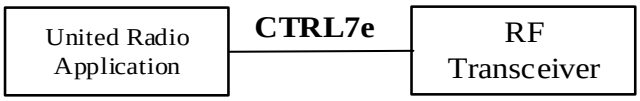

Figure 5.16: Illustration of reference points for RVM protection services

Figure 5.16 illustrates reference point DCTRL3e, which is related to the RVM protection services.

Reference Point **CTRL7e** is an interface between URA and RF transceiver, through which URA requests RF transceiver to select RF protection class and receives a response from RF transceiver.

Reference Point **CTRL7e** is an interface between URA and RF transceiver, through which URA requests RF transceiver to change of RF protection class and receives a response from RF transceiver.

Reference Point **CTRL7e** is an interface between URA and RF transceiver, through which URA requests RF transceiver to get RF protection class status, information on data modification by RF protection, information on cross RAT interference by RF protection, emergency switch off of RF front-end, and receives a response from RF transceiver.

5.13 Reference points required for security functions

The figures below illustrate the required reference points for each security functions.

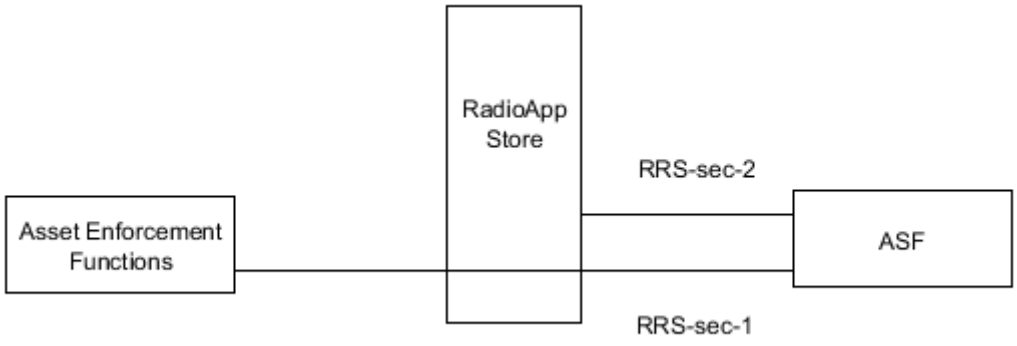

Figure 5.17: Illustration of reference points for protection of RE Configuration Policy, RAP, and other assets

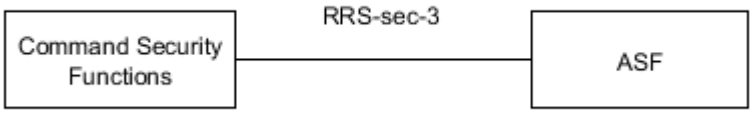

Figure 5.18: Illustration of reference points for protection of configuration enforcement commands

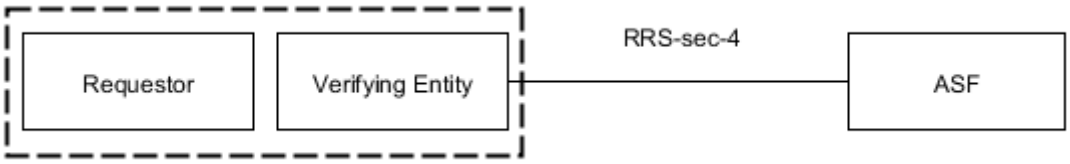

Figure 5.19: Illustration of reference points for remote attestation

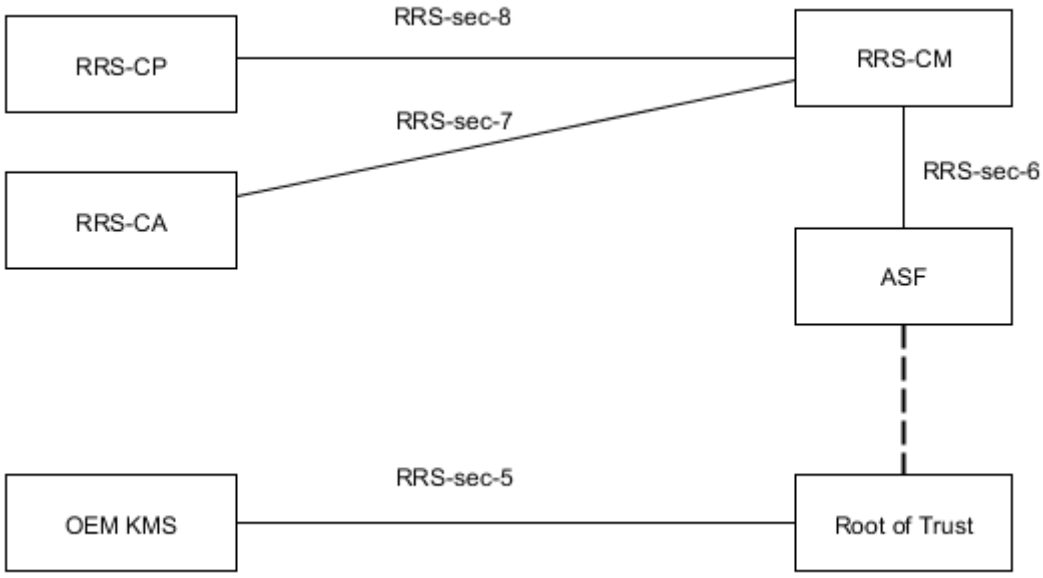

Figure 5.20: Illustration of reference points for long-term management

The reference points are defined as follows:

- Reference Point **RRS-sec-1:** indirect interactions between the ASF and Asset Endorsement Functions for the validation of the RE Configuration Policy, RAP, and other assets (signature verification).
- Reference Point **RRS-sec-2:** direct interactions between the ASF and the RadioApp Store for communication security.
- Reference Point **RRS-sec-3:** direct interactions between the ASF and Command Security Functions for the validation of configuration enforcement commands.
- Reference Point **RRS-sec-4:** direct interactions between the ASF and the Verifying Entity for remote attestation.
- Reference Point **RRS-sec-5:** offline provisioning of secrets by the manufacturer into the Hardware Root of Trust.
- Reference Point **RRS-sec-6:** direct interactions between the RRS-CM and the ASF (providing access to security functions).
- Reference Point **RRS-sec-7:** direct interactions between the RRS-CA and the RRS-CM for transfer of authority and designation of the RRS-CP.
- Reference Point **RRS-sec-8:** direct interactions between the RRS-CP and the RRS-CM for provisioning of the RRS Configuration Profile.

6 Reconfigurable MD high level operating procedures

6.1 Procedures for installation/uninstallation and creating/deleting instance of a URA

Figure 6.1 illustrates a signalling diagram associated with the installation and uninstallation of a URA.

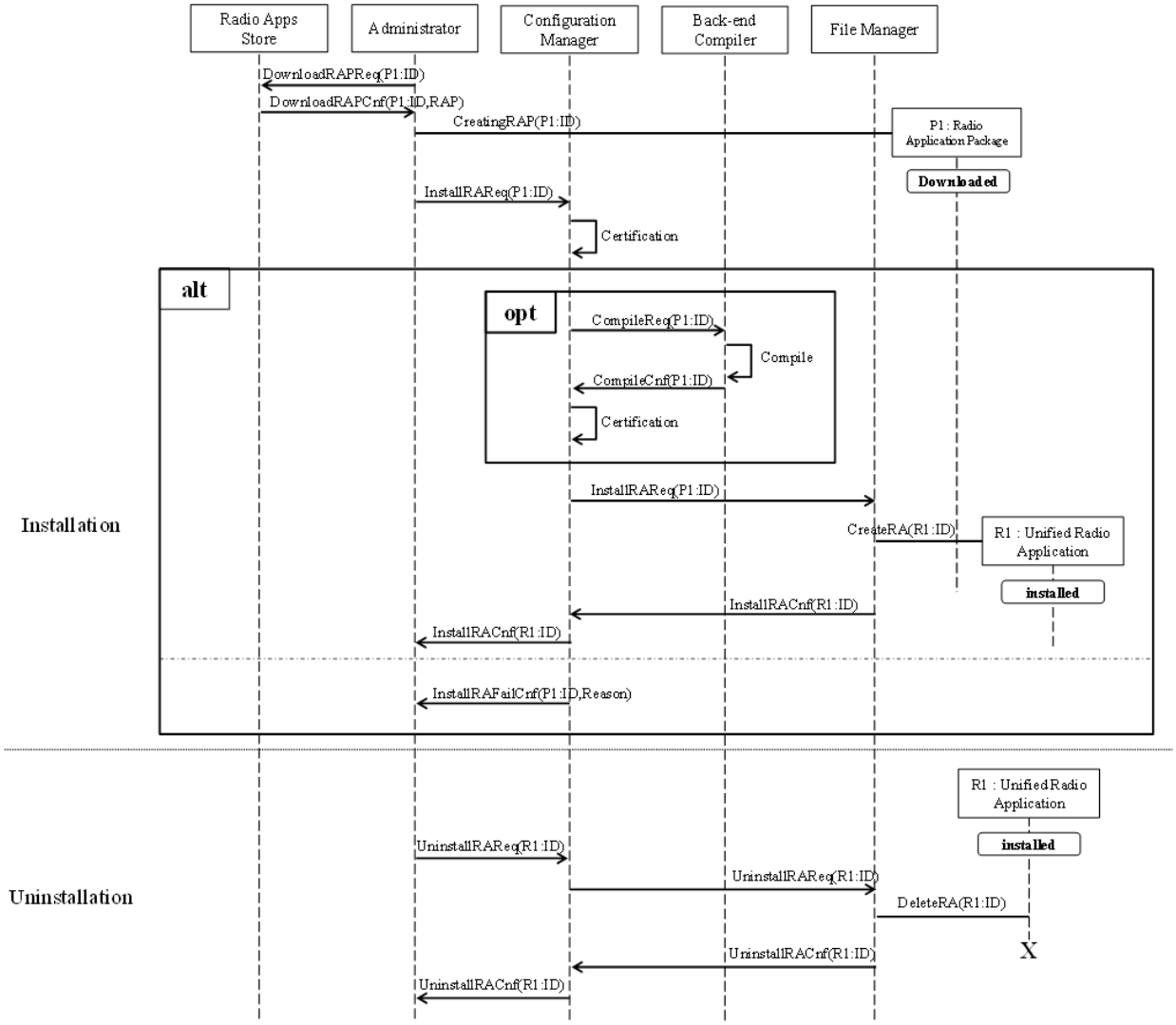

Figure 6.1: URA (un)installation

The procedure can be summarized as follows:

- Administrator sends a *DownloadRAPReq* signal including RAP (P1) identification (ID) to RadioApp Store.
- Administrator receives a *DownloadRAPCnf* signal including RAP (P1) ID and RAP from RadioApp Store.
- Administrator sends an *InstallRAReq* signal including RAP (P1) ID to CM to request URA installation.
- CM first performs the URA code certification procedure in order to verify its compatibility, authentication, etc.
- CM sends the *InstallRAReq* signal including RAP (P1) ID to File Manager (FM) to perform installation of URA.
- FM performs installation of URA and transfers an *InstallRACnf* signal including URA (R1) ID to CM, which transfers the *InstallRACnf* signal including URA (R1) ID to Administrator.
- If the downloaded URA is an IR, CM first sends a *CompileReq* signal including RAP (P1) ID to Back-end Compiler. After completion of back-end compilation, Back-end Compiler transfers a *CompileCnf* signal including RAP (P1) ID to CM, which performs the certification of the back-end compiled URA code. Only after the URA code certification procedure is successfully completed, can the URA installation take place.
- In case of installation failure, CM reports Administrator the failure of URA installation using an *InstallRAFailCnf* signal including RAP (P1) ID and failure reason.

The procedure for the uninstallation is shown in Figure 6.1 and it can be summarized as follows:

- Administrator transfers an *UninstallRAReq* signal including the ID of the URA (R1) to be uninstalled to CM.
- CM sends the *UninstallRAReq* signal including URA (R1) ID to FM.
- FM performs the uninstallation of URA and sends back an *UninstallRACnf* signal including the URA (R1) ID to CM.
- CM sends the *UninstallRACnf* signal including URA (R1) ID to Administrator.

Figure 6.2 illustrates a signalling diagram showing the procedure of creation and deletion of an instance of a URA.

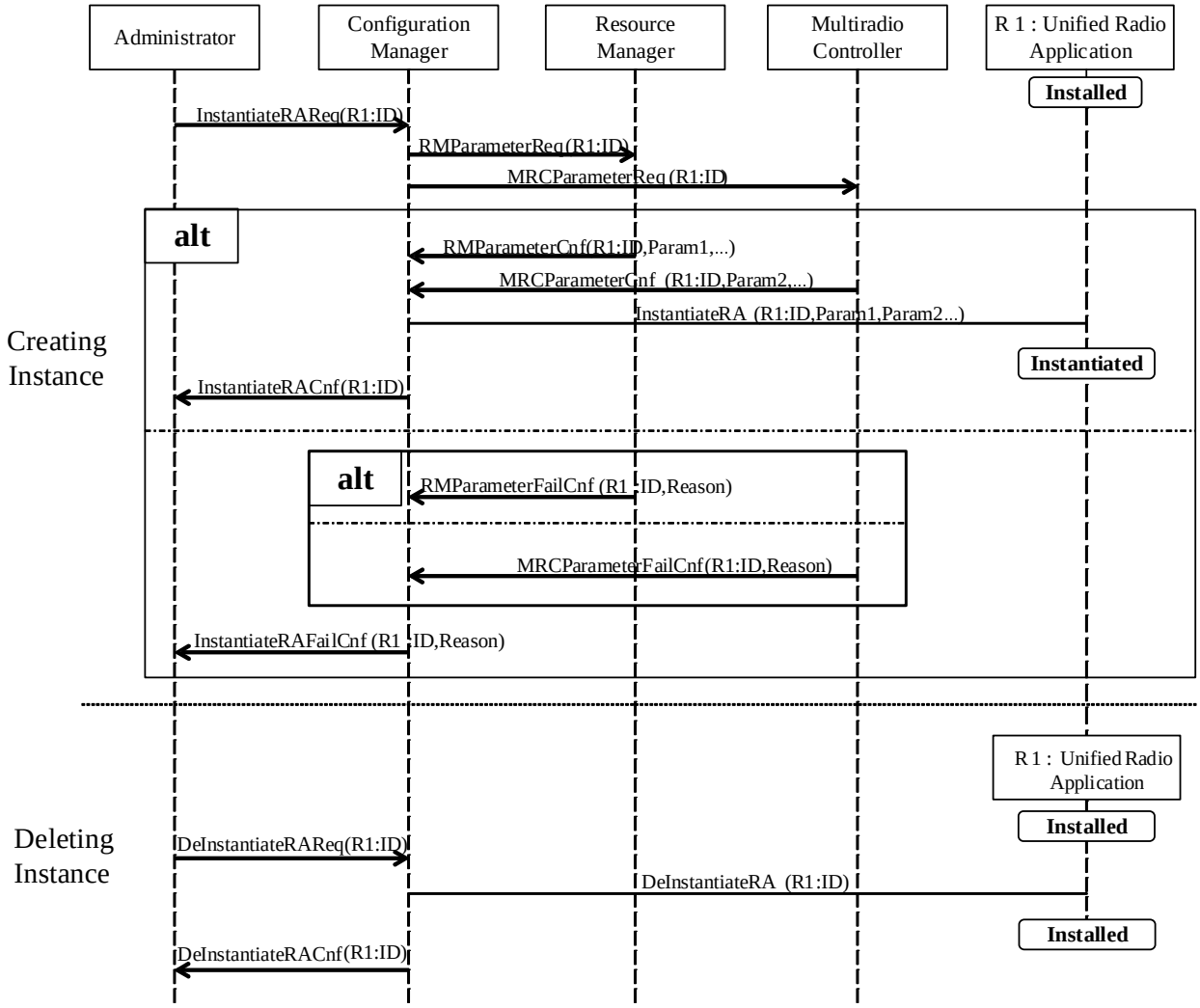

Figure 6.2: URA creation and deletion

The procedure related to the creation of an instance of an URA can be summarized as follows:

- For creating an instance of an installed URA, Administrator transfers an *InstantiateRAReq* signal including the ID of the URA (R1) to be instantiated to CM.
- CM transfers an *RMParameterReq* signal and *an MRCParameterReq* signal including the ID of the URA (R1) in order to get the parameters needed for URA activation (e.g. Forward Error Correction (FEC) parameters, MIMO parameters, bandwidth, etc.) to RM and MRC.
- CM receives an *RMParameterCnf* signal including the ID of the URA (R1) and radio resource parameters from RM.
- CM receives an *MRCParameterCnf* signal including the ID of the URA (R1) and computational resource parameters from MRC.
- CM transfers URA (R1) ID and the received parameters for performing the URA instantiation to ROS.
- After creating an instance, CM transfers an *InstantiateRACnf* signal including URA (R1) ID to Administrator.
- If CM fails to get parameters needed for URA activation from RM and/or MRC, RM and/or MRC reports the failure of parameters transfer to CM using an *RMParameterFailCnf* and/or *MRCParameterFailCnf* signal respectively. In this case CM reports the instantiation failure to Administrator using an *InstantiateRAFailCnf* signal.

The procedure for deleting an instance of an URA can be summarized as follows:

- Administrator transfers the ID of the URA (R1) to be deleted using a *DeInstantiateRAReq* signal to CM.
- Upon request from CM ROS deletes the instance of the designated URA.
- CM acknowledges the completion of the procedure sending a *DeInstantiateRACnf* signal to Administrator.

Figure 6.2a illustrates in more details the security mechanisms involved during RAP provisioning and installation, in particular before the *DownloadRAPReq* and *InstallRAReq* signals as illustrated by Figure 6.1.

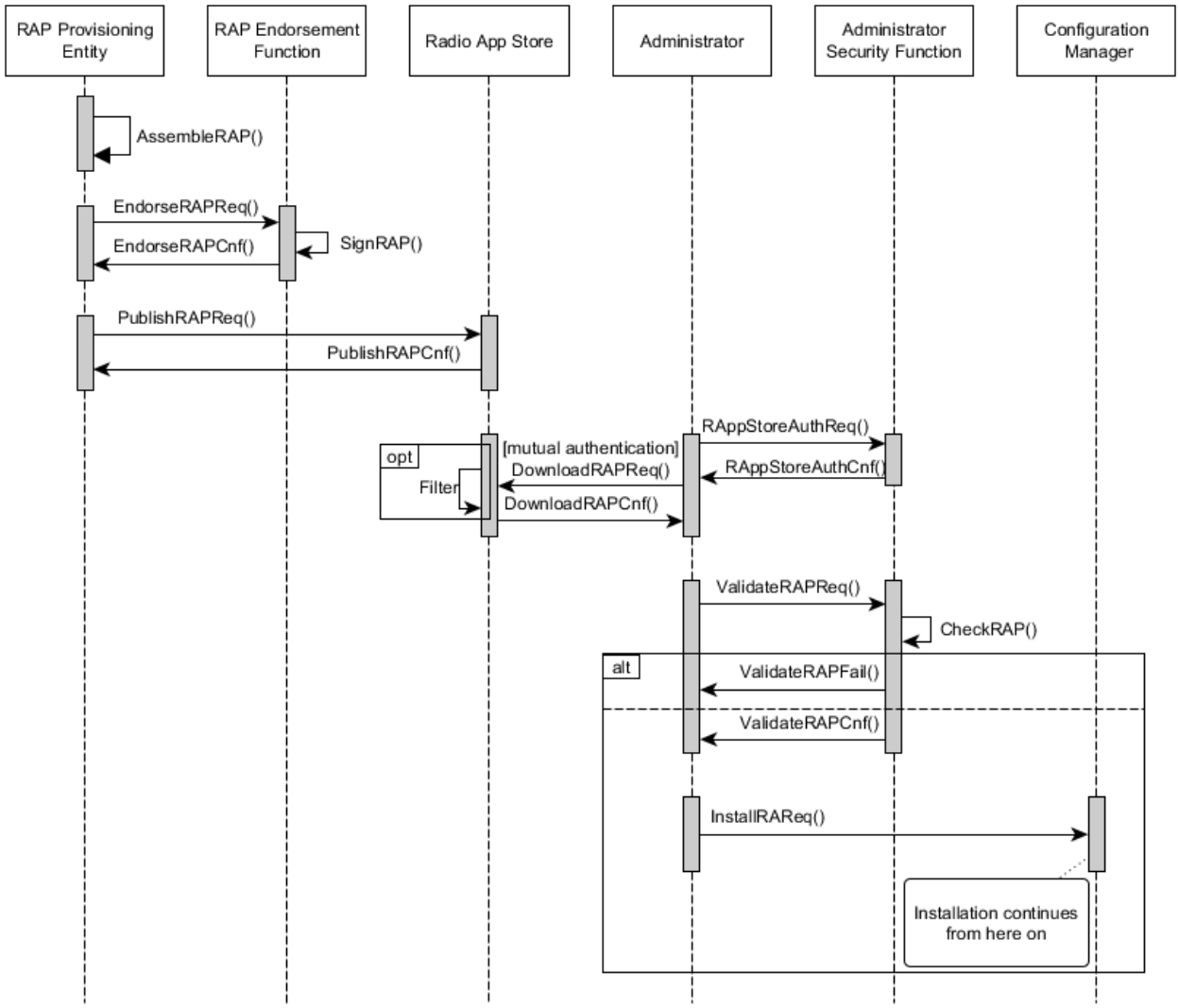

Figure 6.2a: Signalling diagram for RAP endorsement, distribution and validation

The procedure can be summarized as follows:

- The RAP is assembled by the RAP Provisioning Entity (e.g. a Software Manufacturer).
- The RAP is submitted by the RAP Provisioning Entity to the RAP Endorsement Function (a specialization of the Asset Endorsement Functions) via the *EndorseRAPReq()* signal.
- The RAP Endorsement Functions signs the RAP via the reflexive *SignRAP()* signal.
- The RAP Provisioning Entity receives a *EndorseRAPCnf()* signal confirming successful endorsement (signature) of the RAP.
- The RAP Provisioning Entity sends a *PublishRAPReq()* signal to the RadioApp Store for publication of the RAP.
- The RadioApp Store sends a *PublishRAPCnf()* signal to the RAP Provisioning Entity, confirming that the RAP is published and available for download by Reconfigurable MDs.
- The RadioApp Store and the Administrator on the Reconfigurable MD mutually authenticate each other. On the Reconfigurable MD this is done through an exchange of *RAppStoreAuthReq()* and *RAppStoreAuthCnf()* signals between the Administrator and the Administrator Security Function.
- The RadioApp Store may filter the set of available RAPs into a subset suitable for the Reconfigurable MD.
- The Administrator sends a *DownloadRAPReq()* signal to the RadioApp Store in order to download an RAP.
- The RadioApp Store sends a *DownloadRAPCnf()* signal to the Administrator.
- The Administrator sends a *ValidateRAPReq()* signal to the Administrator Security Function in order to verify the digital signature of the RAP. The verification operation is illustrated by the reflexive *CheckRAP()* signal:
  - Upon failure the Administrator Security Function sends a *ValidateRAPFail()* signal to the Administrator and the procedure is aborted.
  - Upon success the Administrator Security Function sends a *ValidateRAPCnf()* signal to the Administrator.
- The Administrator sends a *InstalRAReq()* signal to the Configuration Manager in the Radio Control Framework and the installation procedure continues (refer to Figure 6.1).

NOTE: The upper procedure covers integrity and authentication of RAPs at installation time. Run-time integrity verification is out of scope of the present document and left as an implementation choice.

6.2 Procedures for list checking of URA

Figure 6.3 illustrates a signalling diagram related to the list checking of URA.

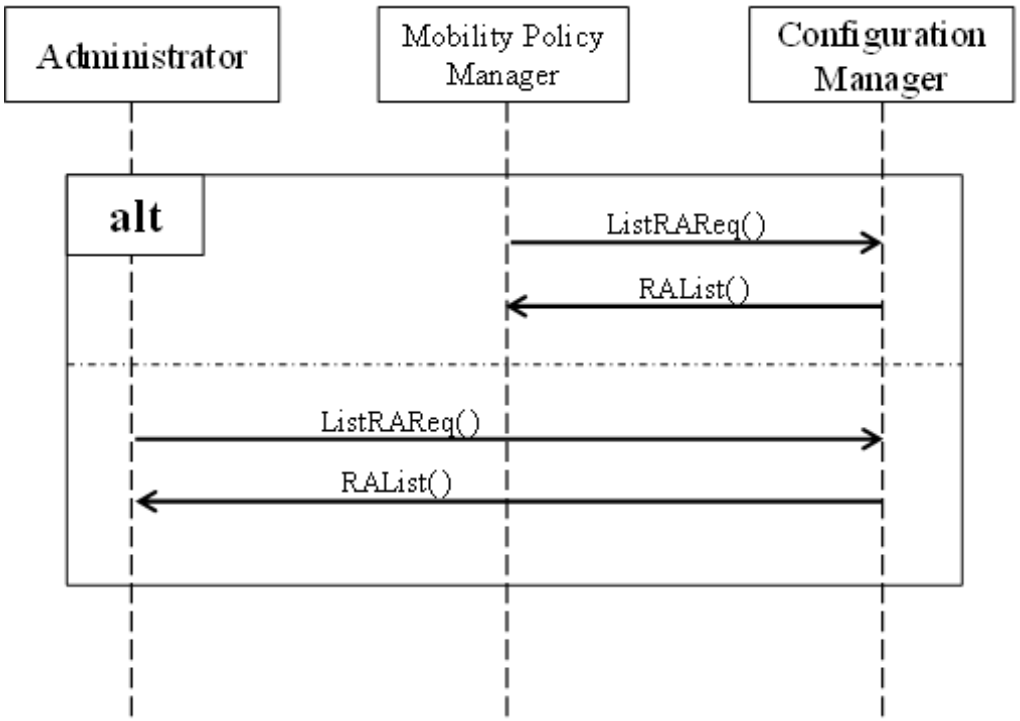

Figure 6.3: URA list checking

The procedure can be summarized as follows:

- Administrator or MPM transfers a *ListRAReq* signal to CM for obtaining the URA list.
- CM transfers the URA list information to Administrator or MPM using an *RAList* signal.

6.3 Procedures for activation/deactivation of URA

Figure 6.4 illustrates a signalling diagram related to the activation of an URA.

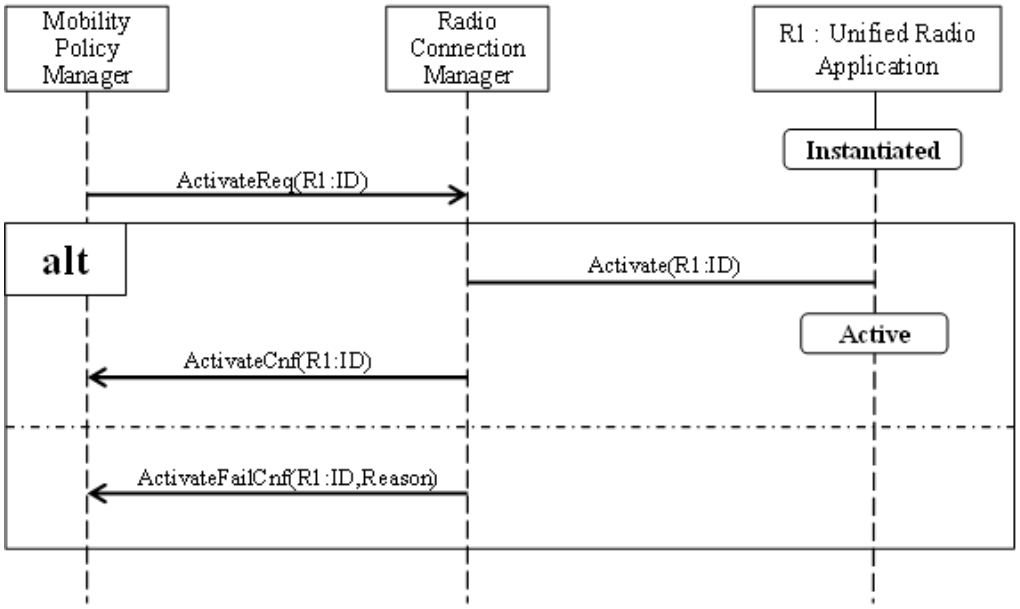

Figure 6.4: URA Activation

The procedure can be summarized as follows:

- MPM transfers an *ActivateReq* signal including the ID of the URA (R1) to RCM.
- Upon request from RCM, ROS activates the designated URA.
- After ROS completes the activation of the URA, RCM sends back to MPM an *ActivateCnf* signal.
- If URA activation is failed, RCM reports the failure to MPM by transferring the failed URA (R1) ID and failure reason in the *ActivateFailCnf* signal.

Figure 6.5 illustrates a signalling diagram related to the deactivation of an URA.

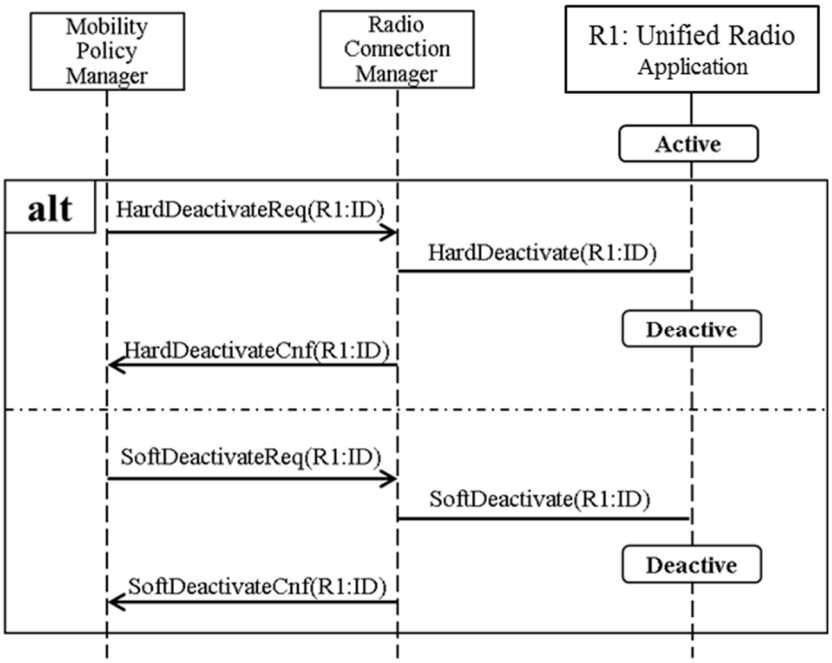

Figure 6.5: URA deactivation

The procedure can be summarized as follows:

- In case of hard deactivation, MPM transfers an *HardDeactivateReq* signal including the ID of the URA (R1) to deactivate to RCM.
- Upon request from RCM, ROS deactivates the designated URA.
- After ROS completes the hard deactivation of the URA, RCM acknowledges the completion of the procedure by sending an *HardDeactivateCnf* signal to MPM.
- In case of soft deactivation, MPM transfers a *SoftDeactivateReq* signal including the ID of the URA (R1) to RCM.
- Upon request from RCM, ROS deactivates the designated URA.
- After ROS completes the soft deactivation of the URA, RCM acknowledges the completion of the procedure by sending a *SoftDeactivateCnf* signal to MPM.

6.4 Procedures for transferring context information

Figure 6.6 illustrates a signalling diagram related to the transfer of context information.

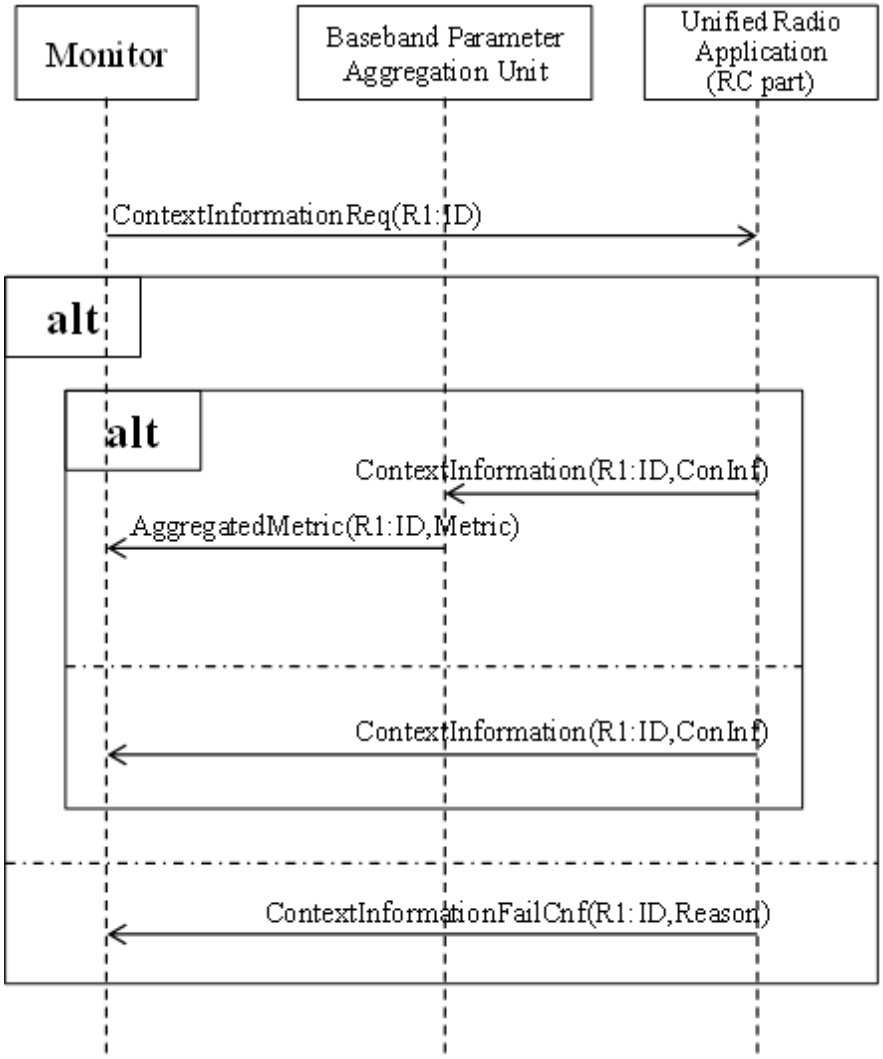

Figure 6.6: Context information transfer

The procedure can be summarized as follows:

- Monitor transfers a *ContextInformationReq* signal including URA (R1) ID to RC in URA.
- RC in URA transfers a *ContextInformation* signal including URA (R1) ID and context information (ConInf) generated in corresponding functional block(s) in URA to Monitor.
- In the case of using a Baseband Parameter Aggregation (BPA) unit [i.2], RC in URA transfers the *ContextInformation* signal including URA (R1) ID and context information (ConInf) to BPA unit. Then the BPA unit aggregates and compresses the context information in order to minimize the bandwidth occupied by the context information to be transferred. Upon completion of the procedure of context information aggregation and compression, BPA unit transfers an *AggregatedMetric* signal including URA (R1) ID and aggregated metric(s) (Metric) to Monitor.
- In the case of generating context information failure, RC in URA transfers a *ContextInformationFailCnf* signal including to URA (R1) ID and failure reason to Monitor.

6.5 Procedure for creating data flow and sending/receiving user data

Figure 6.7 illustrates a signalling diagram related to the creation of a network association.

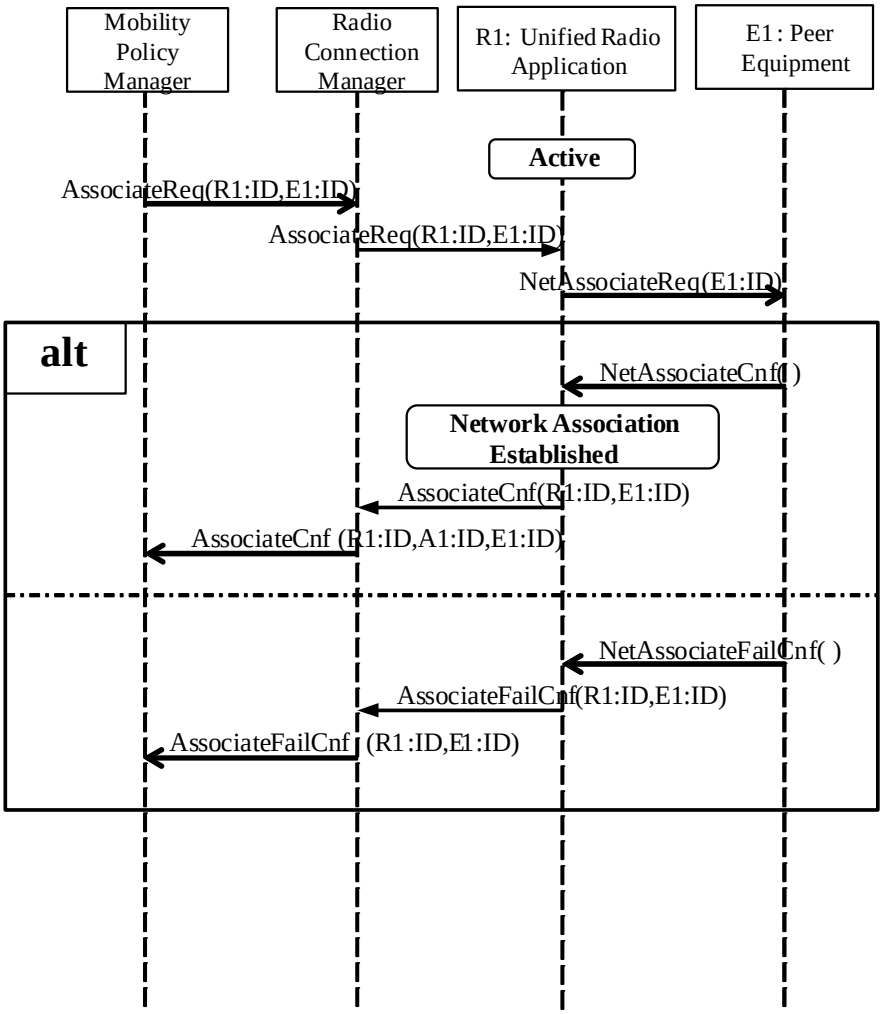

Figure 6.7: Creation of a network association

The procedure can be summarized as follows:

- MPM transfers an *AssociateReq* signal including URA (R1) ID and peer equipment (E1) ID to RCM, where the peer equipment might be Wireless Local Area Network (WLAN) access point(s), Internet Protocol (IP) access node(s) (such as Gateway General Packet Radio Service (GPRS) Support Node (GGSN), etc.) in cellular networks, or Bluetooth headset, digital radio/television broadcasting station(s), Global Positioning System (GPS) satellite(s), etc.

- Upon request from RCM for ROS to create a network association, ROS transfers the *AssociateReq* signal from RCM to URA. Then, URA transfers the ID of corresponding peer equipment (E1) using a *NetAssociateReq* signal.

- Upon completion of the network association creation, peer equipment transfers a *NetAssociateCnf* signal to URA. Then ROS transfers an *AssociateCnf* signal to RCM, which, in turn, transfers it to MPM.

- In the case of a network association failure, peer equipment transfers a *NetAssociateFailCnf* signal to URA. Then ROS transfers an *AssociateFailCnf* signal to RCM, which, in turn, transfers it to MPM.

Figure 6.8a illustrates a signalling diagram related to the creation of a logical radio link association initiated by Mobile Device 1 (D1).

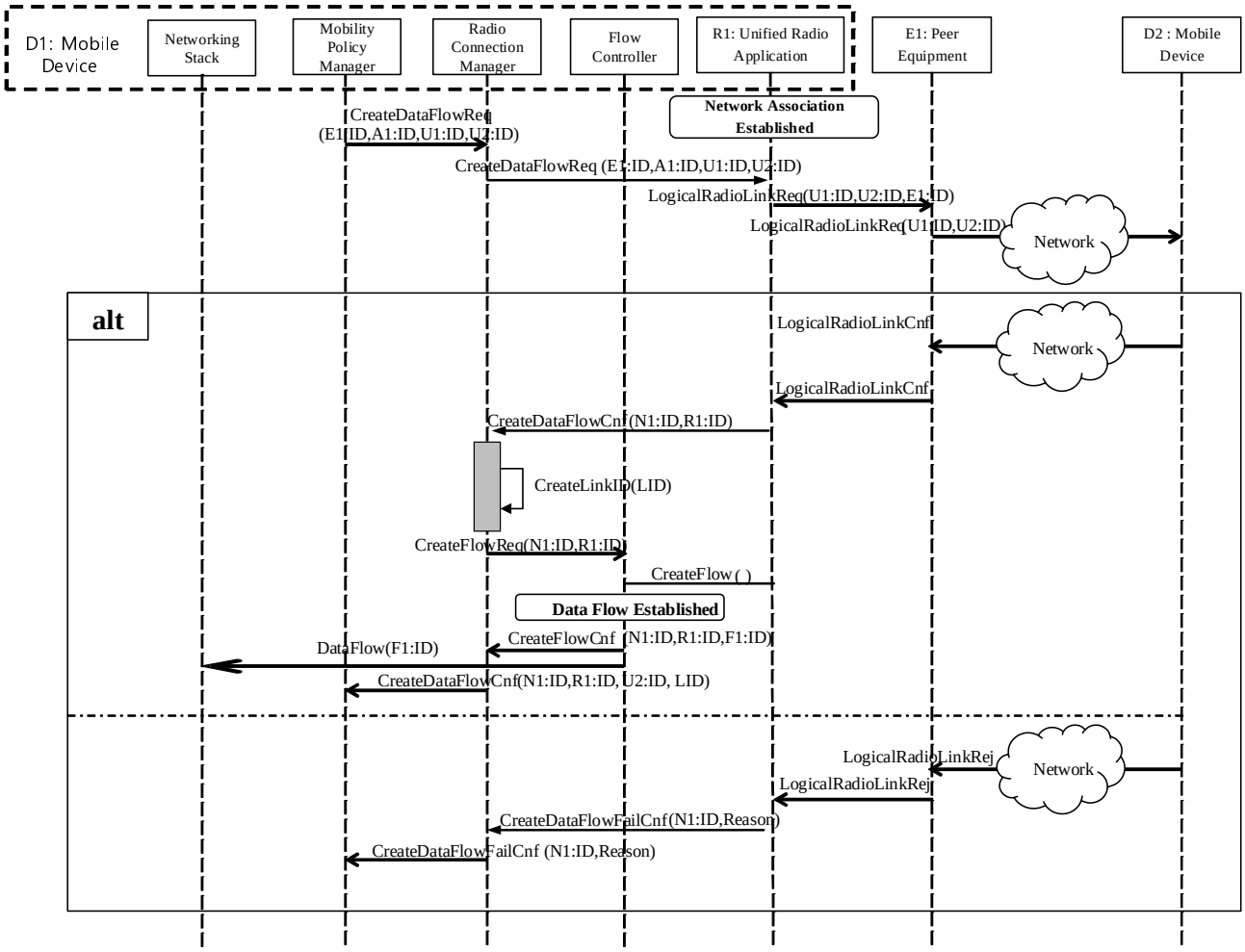

Figure 6.8a: Creation of a logical radio link association initiated by Mobile Device 1 (D1)

The procedure can be summarized as follows:

- MPM transfers a CreateDataFlowReq signal to RCM including peer equipment (E1) ID, active URA (A1) ID, user (U1) ID and the other MD user (U2) ID in order to associate the logical radio link with the other MD.
- RCM requests ROS to create a data flow using the CreateDataFlowReq signal including peer equipment (E1) ID, active URA (A1) ID, user (U1) ID and the other MD user (U2) ID. Then, URA transfers user (U1) ID, the other MD user (U2) ID, and peer equipment (E1) ID using a LogicalRadioLinkReq signal to the peer equipment. Upon receiving the LogicalRadioLinkReq signal, the Network transfers a LogicalRadioLinkCnf signal to peer equipment.

Explanation: User ID identifies Service Access Point where RRS framework provides reconfiguration service. In the case when User ID is associated with Network Layer, User ID denotes a corresponding network (or host, IP) address. When User ID is associated with MAC Layer, it denotes the MAC address. In the case of multicast/broadcast, it will be a multicast/wildcard address.

- Upon transferring the *LogicalRadioLinkCnf* signal from peer equipment to URA, ROS transfers a *CreateDataFlowCnf* signal including network association (N1) ID and URA (R1) ID to RCM. After RCM receives *CreateDataFlowCnf,* RCM creates link (L) ID.
- In order to set up a data flow, RCM transfers a *CreateFlowReq* signal including network association (N1) ID and URA (R1) ID to FC. After creating the data flow, FC transfers a *CreateFlowCnf* signal including network association (N1) ID, URA (R1) ID, and the other MD user (U2) ID to RCM and transfers a *DataFlow* signal including data flow (F1) ID to Networking Stack.

- RCM transfers the *CreateDataFlowCnf* signal including network association (N1) ID, URA (R1) ID, the other MD user (U2) ID, and link (L) ID to MPM.

- In case of failure, when URA receives a *LogicalRadioLinkRej* signal from the peer equipment, ROS transfers a *CreateDataFlowFailCnf* signal including network association (N1) ID and failure reason to RCM, which transfers it to MPM to acknowledge the failure of creating the data flow.

Figure 6.8b illustrates a signalling diagram related to the creation of a logical radio link association initiated by Mobile Device 2 (D2).

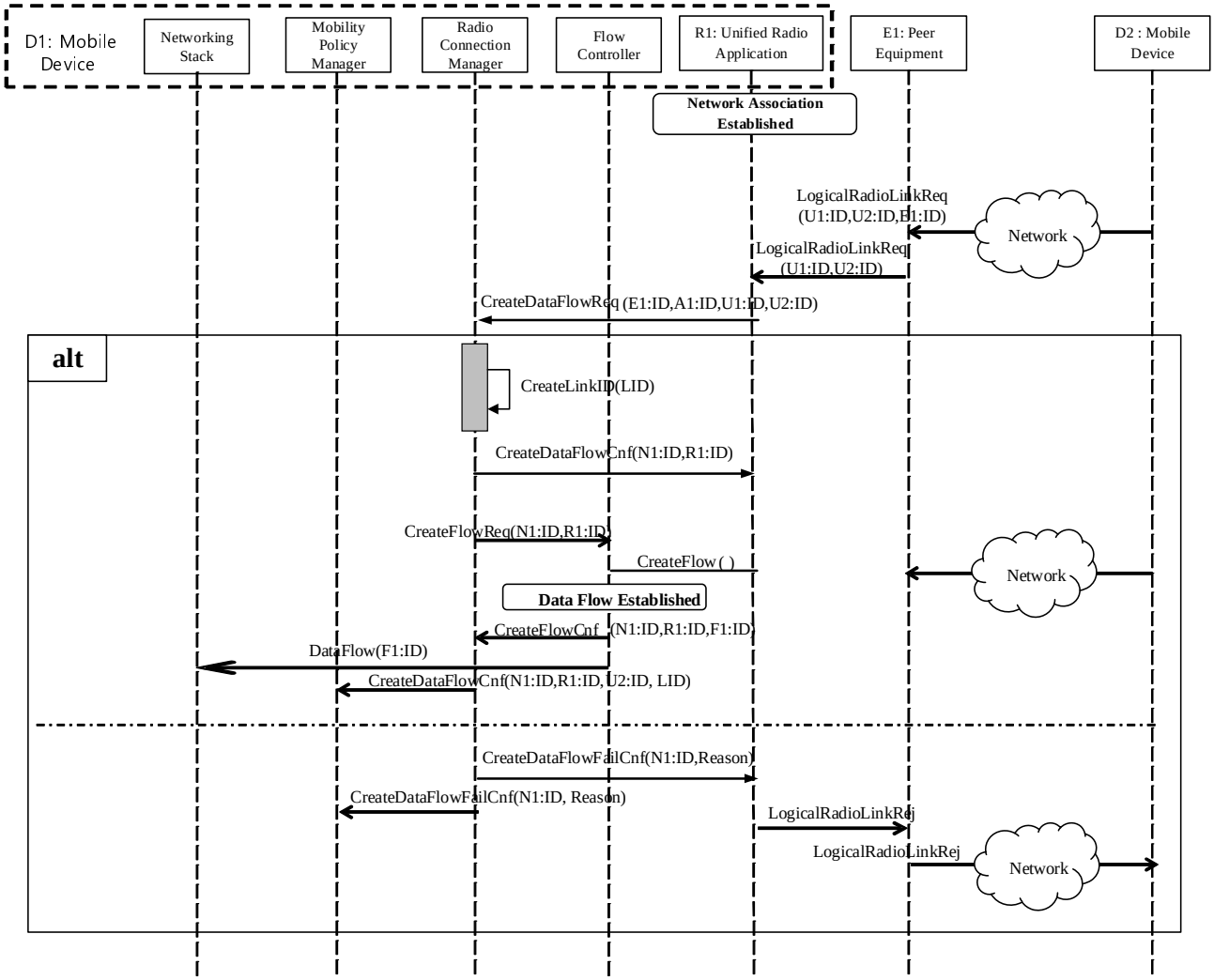

Figure 6.8b: Creation of a logical radio link association initiated by Mobile Device 2 (D2)

The procedure can be summarized as follows:

- Network transfers a *LogicalRadioLinkReq* signal including user (U1) ID, the other MD user (U2) ID, and peer equipment (E1) ID to peer equipment.

- Upon transferring the *LogicalRadioLinkReq* signal including user (U1) ID and the other MD user (U2) ID from peer equipment to URA, URA requests ROS to create a data flow using the *CreateDataFlowReq* signal including peer equipment (E1) ID, active URA (A1) ID, user (U1) ID and the other MD user (U2) ID.

- Upon transferring the *CreateDataFlowReq* signal from URA to RCM, RCM creates link (L) ID. After RCM creates link (L) ID, ROS transfers a *CreateDataFlowCnf* signal including network association (N1) ID and URA (R1) ID to URA.

- In order to set up a data flow, RCM transfers a *CreateFlowReq* signal including network association (N1) ID and URA (R1) ID to FC. After creating the data flow, FC transfers a *CreateFlowCnf* signal including network association (N1) ID, URA (R1) ID, and created data flow (F1) ID to RCM and transfers a *DataFlow* signal including data flow (F1) ID to Networking Stack.

- RCM transfers the *CreateDataFlowCnf* signal including network association (N1) ID, URA (R1) ID, the other MD user (U2) ID, and link (L) ID to MPM.

- In case of failure for creation of a logical radio link association, ROS transfers a *CreateDataFlowFailCnf* signal including network association (N1) ID and failure reason to URA and RCM transfers it to MPM to acknowledge the failure of creating the data flow. URA transfers *LogicalRadioLinkRej* to peer equipment (E1) and upon receiving the *LogicalRadioLinkRej* signal, the Network transfers a *LogicalRadioLinkRej* signal to the other MD.

Figure 6.9 illustrates a signalling diagram of sending data.

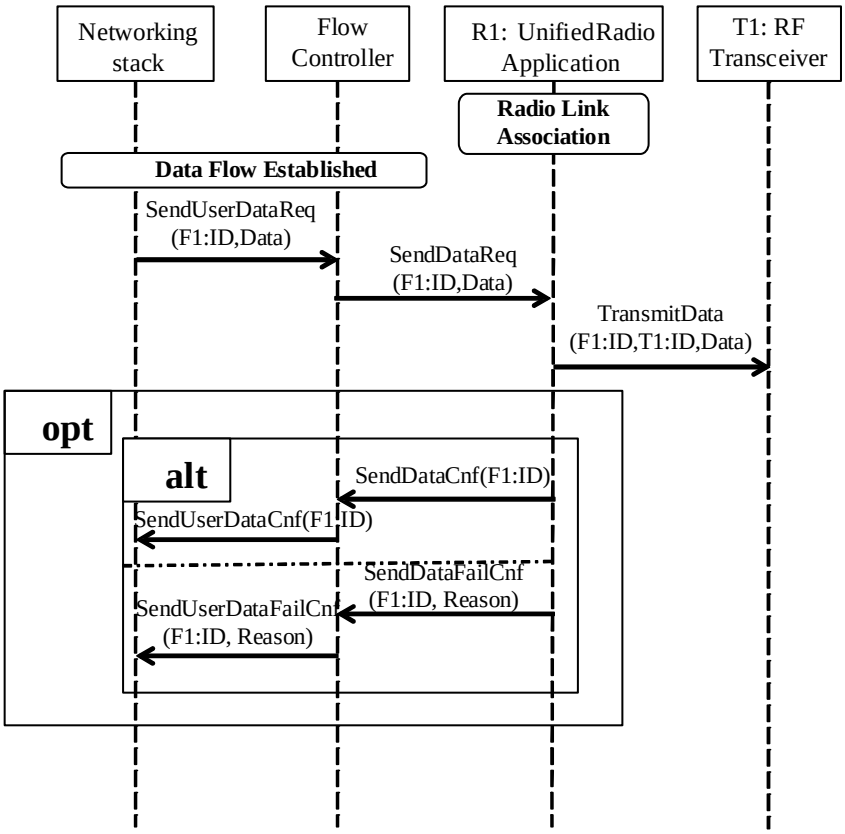

Figure 6.9: Signaling Diagram of sending data

The procedure can be summarized as follows:

- Networking stack transfers a SendUser*DataReq* signal together with data flow (F1) ID and user data to FC in order to send user data.

- Upon receiving the SendUser*DataReq* signal including data flow (F1) ID and user data, FC transfers a *SendDataReq* signal together with data flow (F1) ID and user data to URA.

- URA transfers data flow (F1) ID, RF transceiver (T1) ID, and user data using a *TransmitData* signal to RF transceiver.

- Upon completion of sending data, URA sends back to FC a *SendDataCnf* signal including data flow (F1) ID as an acknowledgement.

- Upon receiving the *SendDataCnf* signal, FC transfers a *SendUserDataCnf* signal together with data flow (F1) ID to Networking stack.

- In the case of data transfer failure, URA reports the failure of sending data to FC by transferring a *SendDataFailCnf* signal including data flow (F1) ID and reason of the failure.
- Upon receiving the *DataFailCnf* signal, FC transfers a *SendUserDataFailCnf* signal together with data flow (F1) ID to Networking stack.

Figure 6.10 illustrates a signalling diagram related to data receiving.

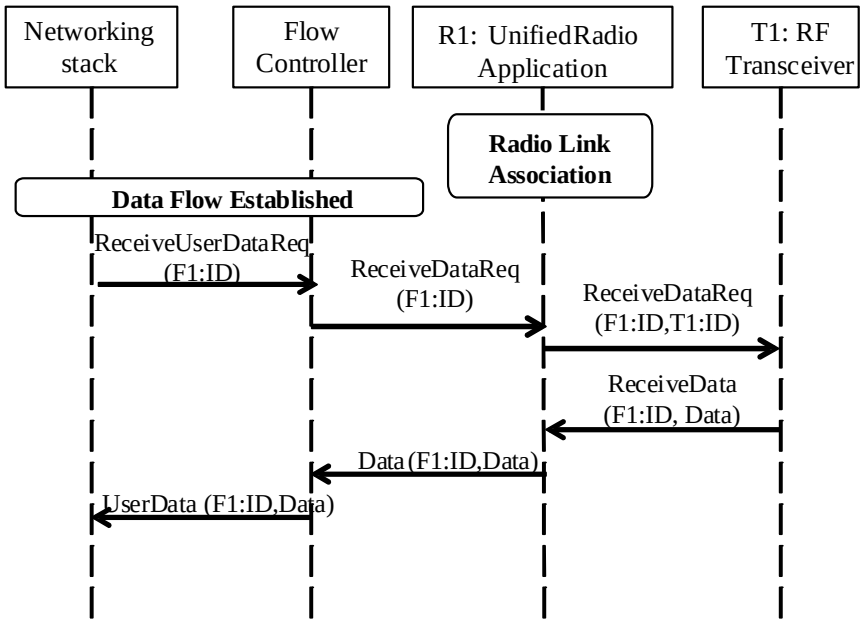

Figure 6.10: Signalling diagram of receiving data

The procedure can be summarized as follows:

- Networking stack transfers a *ReceiveUserDataReq* signal together with data flow (F1) ID to FC in order to receive user data.
- Upon receiving the ReceiveUser*DataReq* signal including data flow (F1) ID, FC transfers a *ReceiveDataReq* signal together with data flow (F1) ID to URA.
- URA transfers data flow (F1) ID and RF transceiver (T1) ID using a *ReceiveDataReq* signal to RF transceiver.
- RF transceiver transfers ReceiveData including data flow (F1) ID and user data to URA.
- URA transfers a *Data* signal including data flow (F1) ID and user data to FC after receiving the data from RF transceiver.
- FC transfers a *UserData* signal including data flow (F1) ID and user data received from URA to Networking stack.

6.6 Procedures for radio environment measurements

Figure 6.11 illustrates a signalling diagram related to the radio environment measurements.

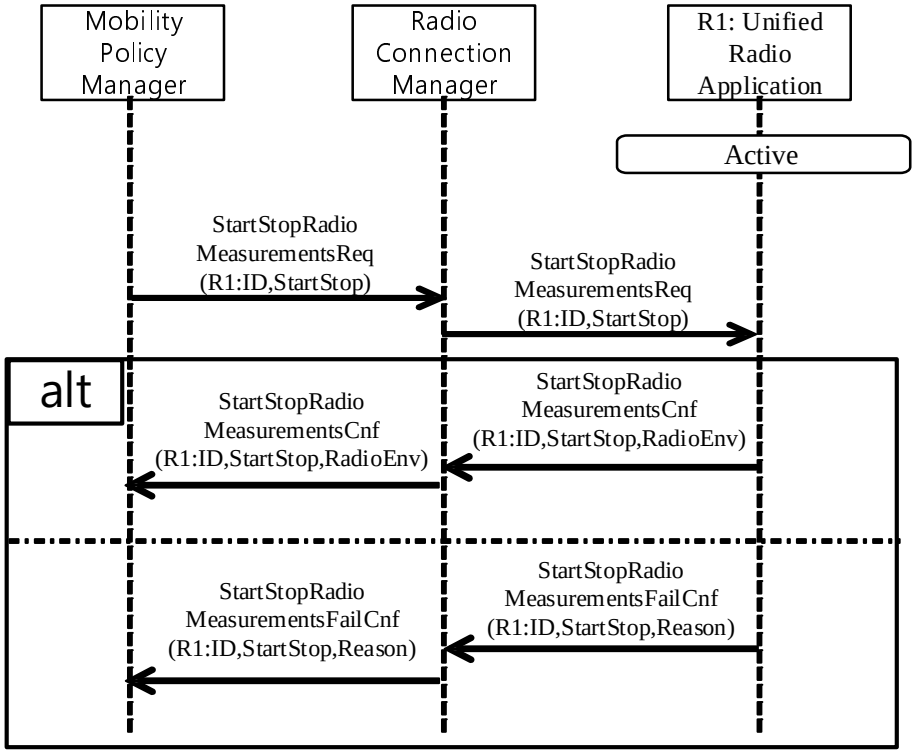

Figure 6.11: Radio environment measurements

The procedure can be summarized as follows:

- MPM transfers a *StartStop RadioMeasurementsReq* signal including URA (R1) ID and start/stop configuration to RCM. Then, RCM transfers a *StartStopRadioMeasurementsReq* including URA (R1) ID and start/stop configuration (StartStop) to URA.

- Upon completion of the start/stop radio environment measurement, URA transfers a *StartStopRadioMeasurementsCnf* signal including URA (R1) ID, start/stop configuration (StartStop), and radio measurement to RCM. Then, RCM transfers a *StartStopRadioMeasurementsCnf* signal including URA (R1) ID, start/stop configuration (StartStop), and radio measurement (RadioEnv) to MPM.

- In the case of the start/stop radio environment measurement failure, URA transfers a *Start/StopRadioMeasurementsFailCnf* signal including URA (R1) ID, start/stop configuration (StartStop), and failure reason to RCM. Then, RCM transfers a *Start/StopRadioMeasurementsFailCnf* signal including URA (R1) ID, start/stop configuration (StartStop), and failure reason to MPM.

6.7 Procedure for reporting discovered peer equipment

Figure 6.12 illustrates a signalling diagram related to report discovered peer equipments.

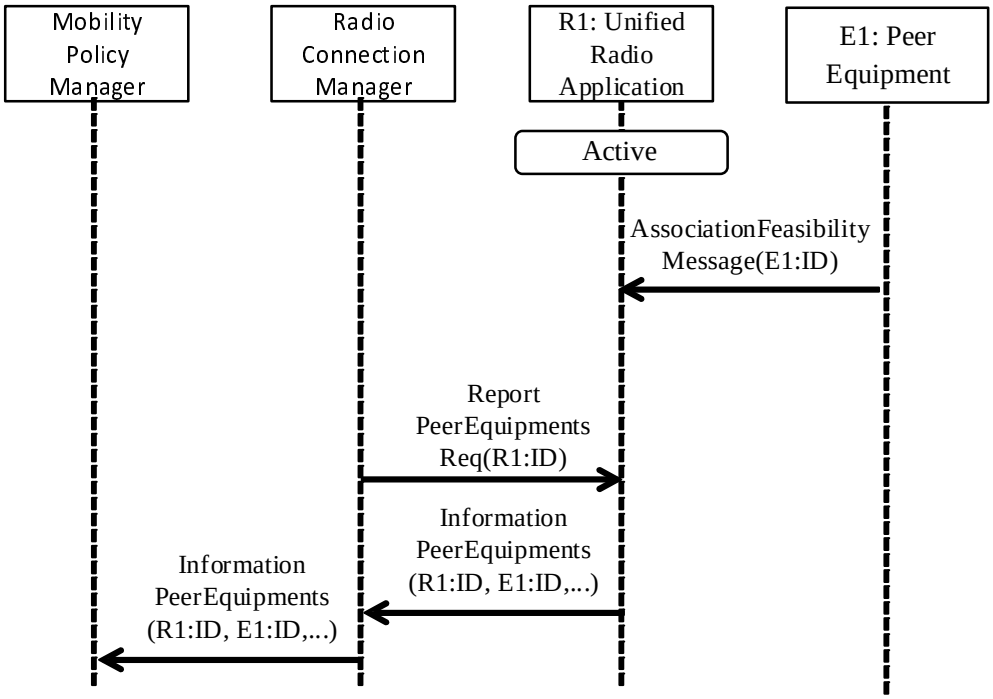

Figure 6.12: Report of discovered peer equipment

The procedure can be summarized as follows:

- Peer equipment transfers a *AssociationFeasibilityMessage* signal including peer equipment (E1) ID to URA.
- RCM transfers a *ReportPeerEquipmentsReq* signal including URA (R1) ID to URA.
- URA transfers a *InformationPeerEquipments* signal including URA (R1) ID and peer equipment (E1) ID to RCM.
- RCM transfers a *InformationPeerEquipments* signal including URA (R1) ID and peer equipment (E1) ID to MPM.

6.8 Procedure for flexible data flow

Figure 6.13 illustrates a signalling diagram related to the flexible data flow.

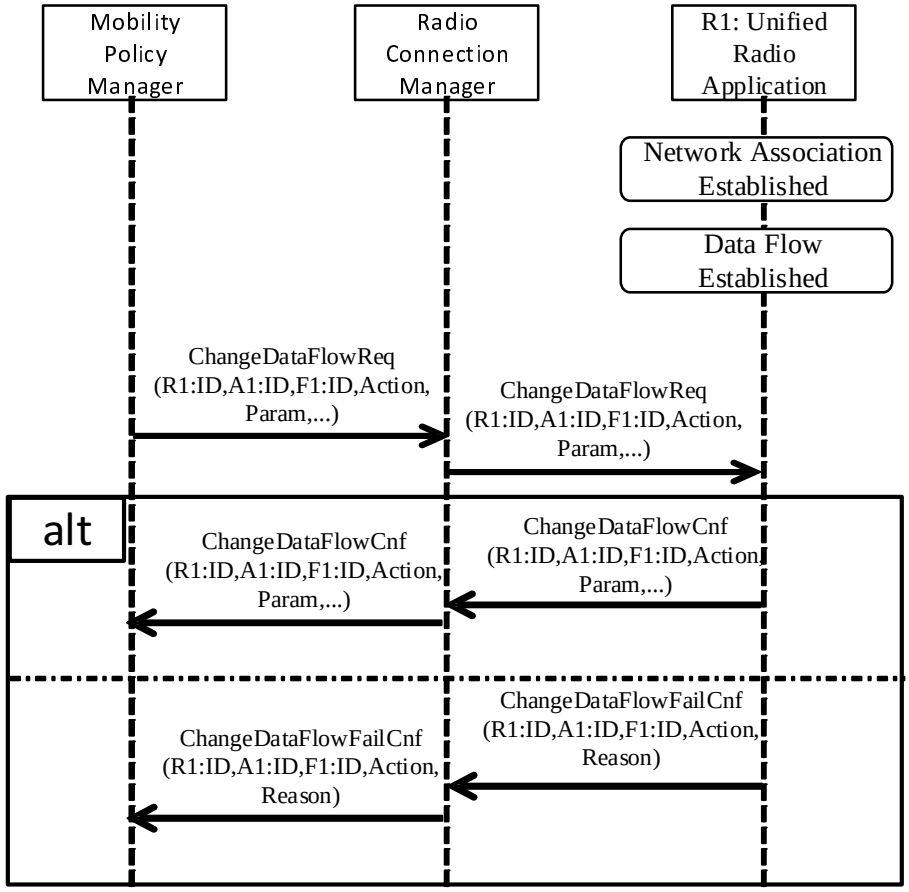

Figure 6.13: Flexible data flow

The procedure can be summarized as follows:

- MPM transfers a *ChangeDataFlowReq* signal including URA (R1) ID, active URA (A1) ID, data flow 1 (F1) ID, action configuration such as move/partition/combine of the data flow (Action), and computational resource parameters (Param) to RCM.
- RCM transfers a *ChangeDataFlowReq* signal including URA (R1) ID, active URA (A1) ID, data flow 1 (F1) ID, action configuration (Action), and computational resource parameters (Param) to URA.
- Upon completion of changing data flow, URA transfers a *ChangeDataFlowCnf* signal including URA (R1) ID, active URA (A1) ID, data flow 1 (F1) ID, action configuration (Action), and computational resource parameters (Param) to RCM. Then, RCM transfers *ChangeDataFlowCnf* signal including URA (R1) ID, active URA (A1) ID, data flow 1 (F1) ID, action configuration (Action), and computational resource parameters (Param) to MPM.
- In the case of changing data flow failure, URA transfers a *ChangeDataFlowFailCnf* signal including URA (R1) ID, active URA (A1) ID, data flow 1 (F1) ID, action configuration (Action), and failure reason to RCM. Then, RCM transfers *ChangeDataFlowFailCnf* signal including URA (R1) ID, active URA (A1) ID, data flow 1 (F1) ID, action configuration (Action), and failure reason to MPM.

6.9 Procedure for data flow control

Figure 6.14 illustrates a signalling diagram related to the data flow control.

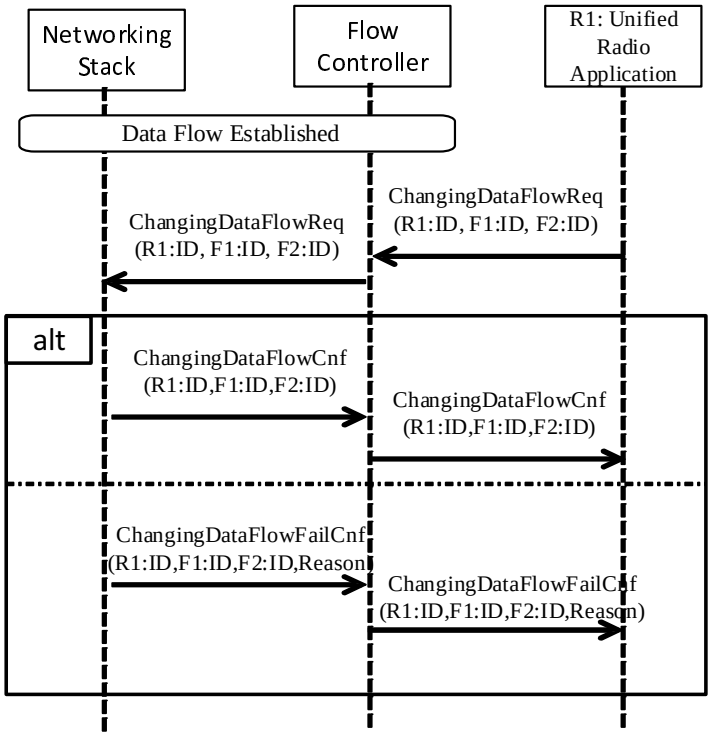

Figure 6.14: Data flow control

The procedure can be summarized as follows:

- URA transfers a *ChangingDataFlowReq* signal including URA (R1) ID, data flow 1 (F1) ID, and data flow 2 (F2) ID to FC.
- FC transfers a *ChangingDataFlowReq* signal including URA (R1) ID, data flow 1 (F1) ID, and data flow 2 (F2) ID to Networking stack.
- Upon completion of data flow configuration changes, Networking stack transfers a *ChangingDataFlowCnf* signal including URA (R1) ID, data flow 1 (F1) ID, and data flow 2 (F2) ID to FC. Then FC transfers a *ChangingDataFlowCnf* signal to URA.
- In the case of a data flow configuration changes failure, Networking stack transfers a *ChangingDataFlowFailCnf* signal including URA (R1) ID, data flow 1 (F1) ID, and data flow 2 (F2) ID to FC. Then FC transfers *ChangingDataFlowFailCnf* signal to URA.

6.10 Procedure for synchronizing radio time

Figure 6.15 illustrates a signalling diagram related to the synchronizing radio time.

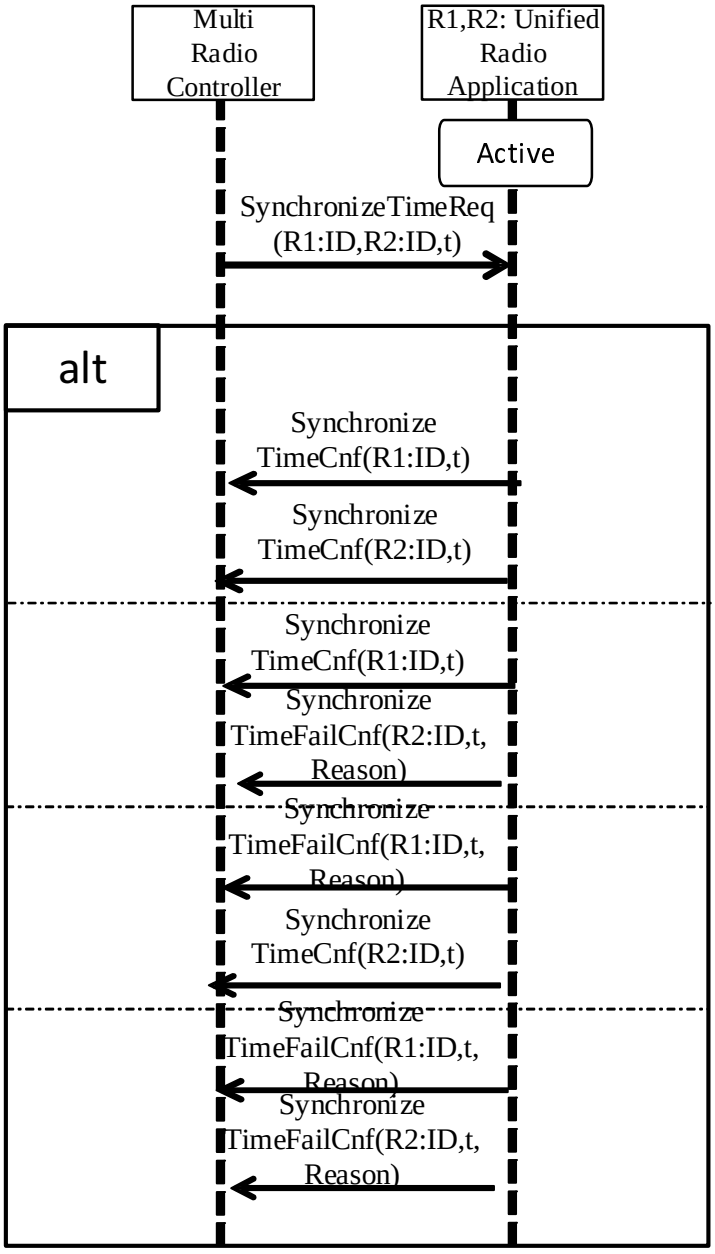

Figure 6.15: Synchronizing radio time

The procedure can be summarized as follows:

- MRC transfers a *SynchronizeTimeReq* signal including URA1 (R1) ID, URA2 (R2) ID, and synchronize radio time (t) to URA1 and URA2.
- Upon completion of synchronizing radio time at both URA1 and URA2, URA1 transfers a *SynchronizeTimeCnf* signal including URA1 (R1) ID and synchronize radio time (t) to MRC. Then URA 2 transfers a *SynchronizeTimeCnf* signal including URA2 (R2) ID and synchronize radio time (t) to MRC.
- In the case of completion of synchronizing radio time at URA1 and failure of synchronizing radio time at URA2, URA1 transfers a *SynchronizeTimeCnf* signal including URA1 (R1) ID and synchronize radio time (t) to MRC. Then URA 2 transfers a *SynchronizeTimeFailCnf* signal including URA2 ID, synchronize radio time (t), and failure reason to MRC.
- In the case of failure of synchronizing radio time at URA1 and completion of synchronizing radio time at URA2, URA1 transfers a *SynchronizeTimeFailCnf* signal including URA1 (R1) ID, synchronize radio time (t), and failure reason to MRC. Then URA 2 transfers a *SynchronizeTimeCnf* signal including URA2 (R2) ID and synchronize radio time (t) to MRC.
- In the case of failure of synchronizing radio time at URA1 and failure of synchronizing radio time at URA2, URA1 transfers a *SynchronizeTimeFailCnf* signal including URA1 (R1) ID, synchronize radio time (t), failure reason to MRC. Then URA 2 transfers a *SynchronizeTimeFailCnf* signal including URA2 (R2) ID, synchronize radio time (t), failure reason to MRC.

6.11 Procedure for control of reconfigurable RF transceiver

Figure 6.16 illustrates a signalling diagram related to the spectrum control services.

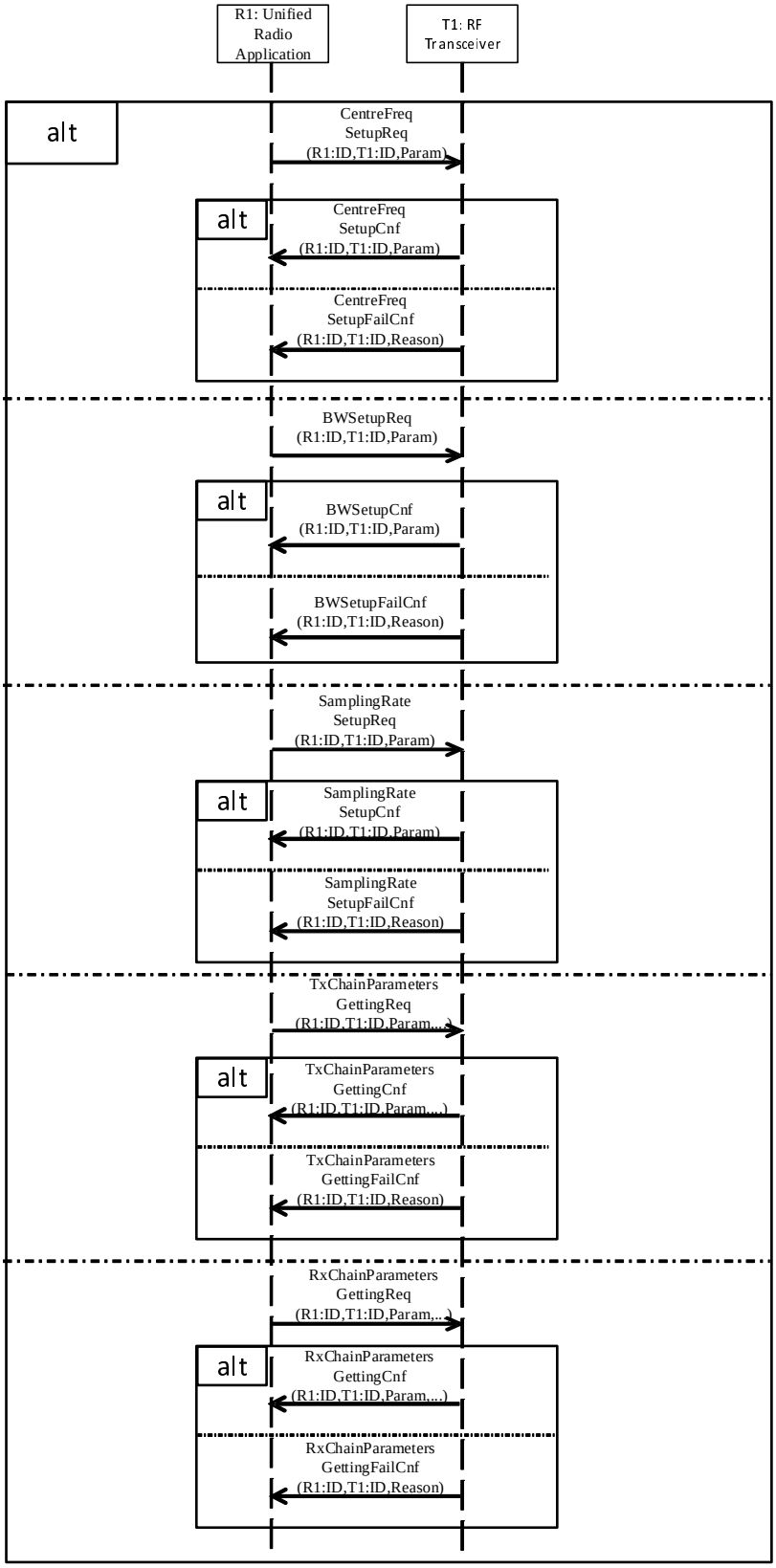

Figure 6.16: Spectrum control services

The procedure can be summarized as follows:

- In the case of request of centre frequency, URA transfers a *CentreFreqSetupReq* signal including URA (R1) ID, RF Transceiver (T1) ID, and centre frequency parameter to RF transceiver.
  - Upon completion of request of centre frequency, URA transfers a *CentreFreqSetupCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and centre frequency parameter to RF transceiver.
  - In the case of request of centre frequency failure, URA transfers a *CentreFreqSetupCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.
- In the case of request of bandwidth, URA transfers a *BWSetupReq* signal including URA (R1) ID, RF Transceiver (T1) ID, and bandwidth parameter to RF transceiver.
  - Upon completion of request of bandwidth, URA transfers a *BWSetupCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and bandwidth parameter to RF transceiver.
  - In the case of request of centre frequency failure, URA transfers a *BWSetupFailCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.
- In the case of request of sampling rate, URA transfers a *SamplingRateSetupReq* signal including URA (R1) ID, RF Transceiver (T1) ID, and sampling rate parameter to RF transceiver.
  - Upon completion of request of sampling rate, URA transfers a *SamplingRateSetupCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and sampling rate parameter to RF transceiver.
  - In the case of request of sampling rate failure, URA transfers a *SamplingRateSetupFailCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.
- In the case of request of Tx Chain Parameters, URA transfers a *TxChainParametersGettingReq* signal including URA (R1) ID, RF Transceiver (T1) ID, and chain parameter to RF transceiver.
  - Upon completion of request of Tx Chain Parameters, URA transfers a *TxChainParametersGettingCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and chain parameter to RF transceiver.
  - In the case of request of Tx Chain Parameters failure, URA transfers a *TxChainParametersGettingFailCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.
- In the case of request of Rx Chain Parameters, URA transfers a *RxChainParametersGettingReq* signal including URA (R1) ID, RF Transceiver (T1) ID, and chain parameter to RF transceiver.
  - Upon completion of request of Rx Chain Parameters, URA transfers a *RxChainParametersGettingCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and chain parameter to RF transceiver.
  - In the case of request of Rx Chain Parameters failure, URA transfers a *RxChainParametersGettingFailCnf* signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.

Figure 6.17 illustrates a signalling diagram related to the power control services.

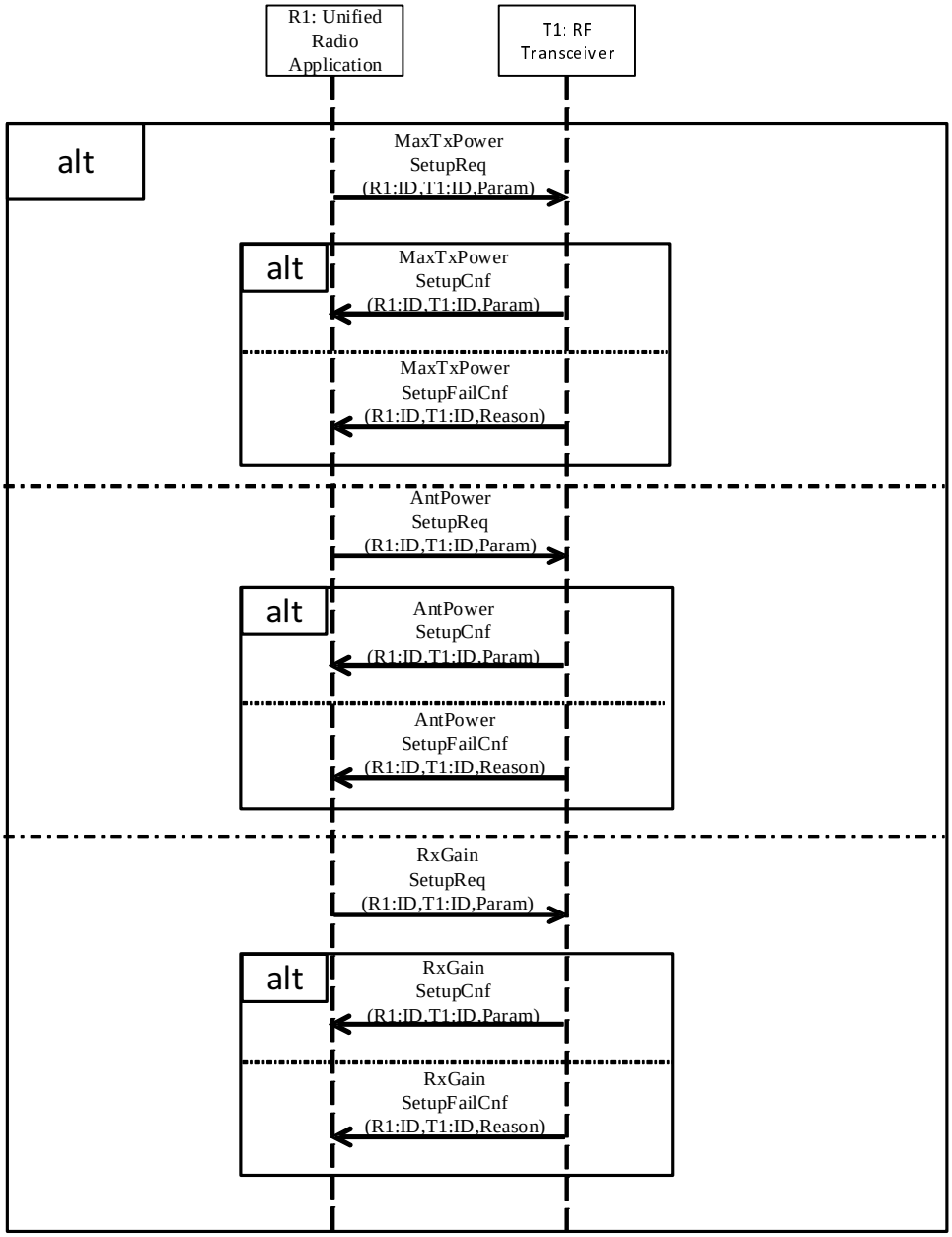

Figure 6.17: Power control services

The procedure can be summarized as follows:

- In the case of request of max tx power, URA transfers a *MaxTxPowerSetupReq* signal including URA (R1) ID, RF Transceiver (T1) ID, and max tx power parameter to RF transceiver.
  - Upon completion of request of max tx power, URA transfers a MaxTxPowerSetupCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and max tx power parameter to RF transceiver.
  - In the case of request of max tx power failure, URA transfers a MaxTxPowerSetupFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.
- In the case of request of antenna power, URA transfers a AntPowerSetupReq signal including URA (R1) ID, RF Transceiver (T1) ID, and antenna power parameter to RF transceiver.
  - Upon completion of request of antenna power, URA transfers a AntPowerSetupCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and antenna power parameter to RF transceiver.
  - In the case of request of antenna power failure, URA transfers a AntPowerSetupFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.

- In the case of request of rx gain, URA transfers a RxGainSetupReq signal including URA (R1) ID, RF Transceiver (T1) ID, and rx gain parameter to RF transceiver.
    - Upon completion of request of rx gain, URA transfers a RxGainSetupCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and rx gain parameter to RF transceiver.
    - In the case of request of rx gain failure, URA transfers a RxGainSetupFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.

Figure 6.18 illustrates a signalling diagram related to the antenna management services.

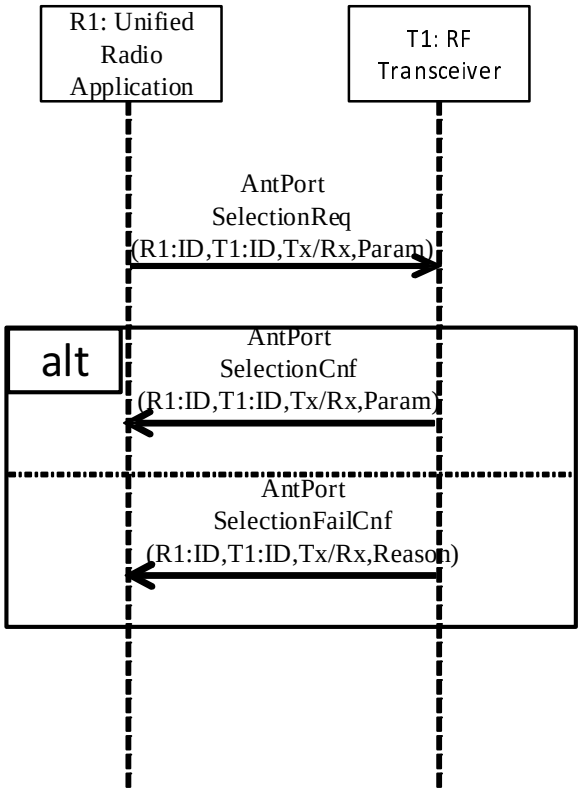

Figure 6.18: Antenna management services

The procedure can be summarized as follows:

- In the case of request of antenna port selection, URA transfers a AntPortSelectionReq signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and antenna port parameter to RF transceiver.
    - Upon completion of request of antenna port selection, URA transfers a AntPortSelectionCnf signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and antenna port parameter to RF transceiver.
    - In the case of request of antenna port selection failure, URA transfers a AntPortSelectionFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and failure reason to RF transceiver.

Figure 6.19 illustrates a signalling diagram related to the Tx/Rx chain control services.

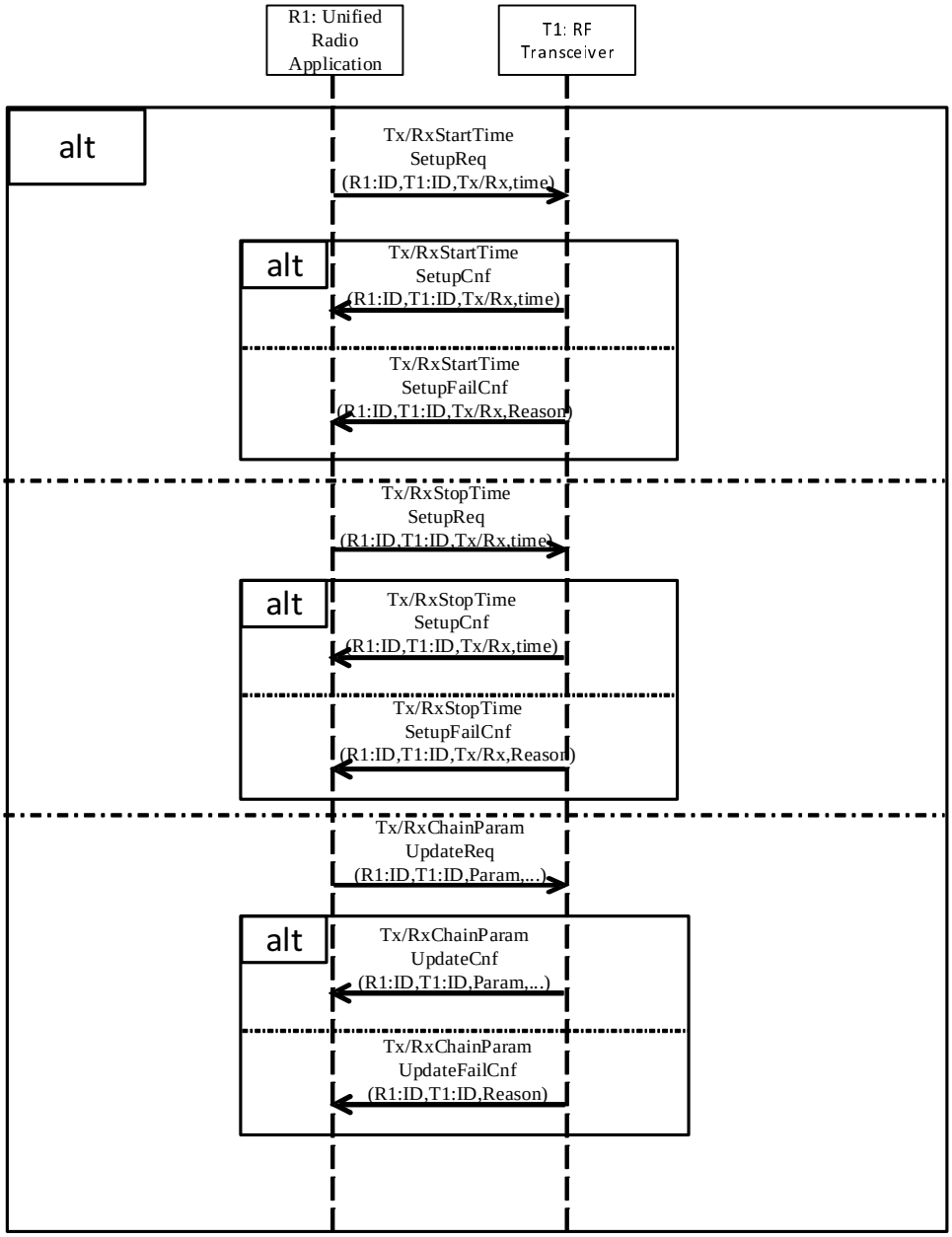

Figure 6.19: Tx/Rx chain control services

The procedure can be summarized as follows:

- In the case of request of Tx/Rx start time, URA transfers a Tx/RxStartTimeSetupReq signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and time to RF transceiver.
  - Upon completion of request of Tx/Rx start time, URA transfers a Tx/RxStartTimeSetupCnf signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and time to RF transceiver.
  - In the case of request of Tx/Rx start time, URA transfers a Tx/RxStartTimeSetupFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and failure reason to RF transceiver.
- In the case of request of Tx/Rx stop time, URA transfers a Tx/RxStopTimeSetupReq signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and time to RF transceiver.
  - Upon completion of request of Tx/Rx stop time, URA transfers a Tx/RxStopTimeSetupCnf signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and time to RF transceiver.
  - In the case of request of Tx/Rx start time, URA transfers a Tx/RxStopTimeSetupFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, Tx/Rx configuration, and failure reason to RF transceiver.

- In the case of request of chain parameter update, URA transfers a Tx/RxChainParamUpdateReq signal including URA (R1) ID, RF Transceiver (T1) ID, and computational resource parameters to RF transceiver.
    - Upon completion of request of Tx/Rx stop time, URA transfers a Tx/RxChainParamUpdateCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and computational resource parameters to RF transceiver.
    - In the case of request of Tx/Rx start time, URA transfers a Tx/RxChainParamUpdateFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.

Figure 6.20 illustrates a signalling diagram related to the RVM protection services.

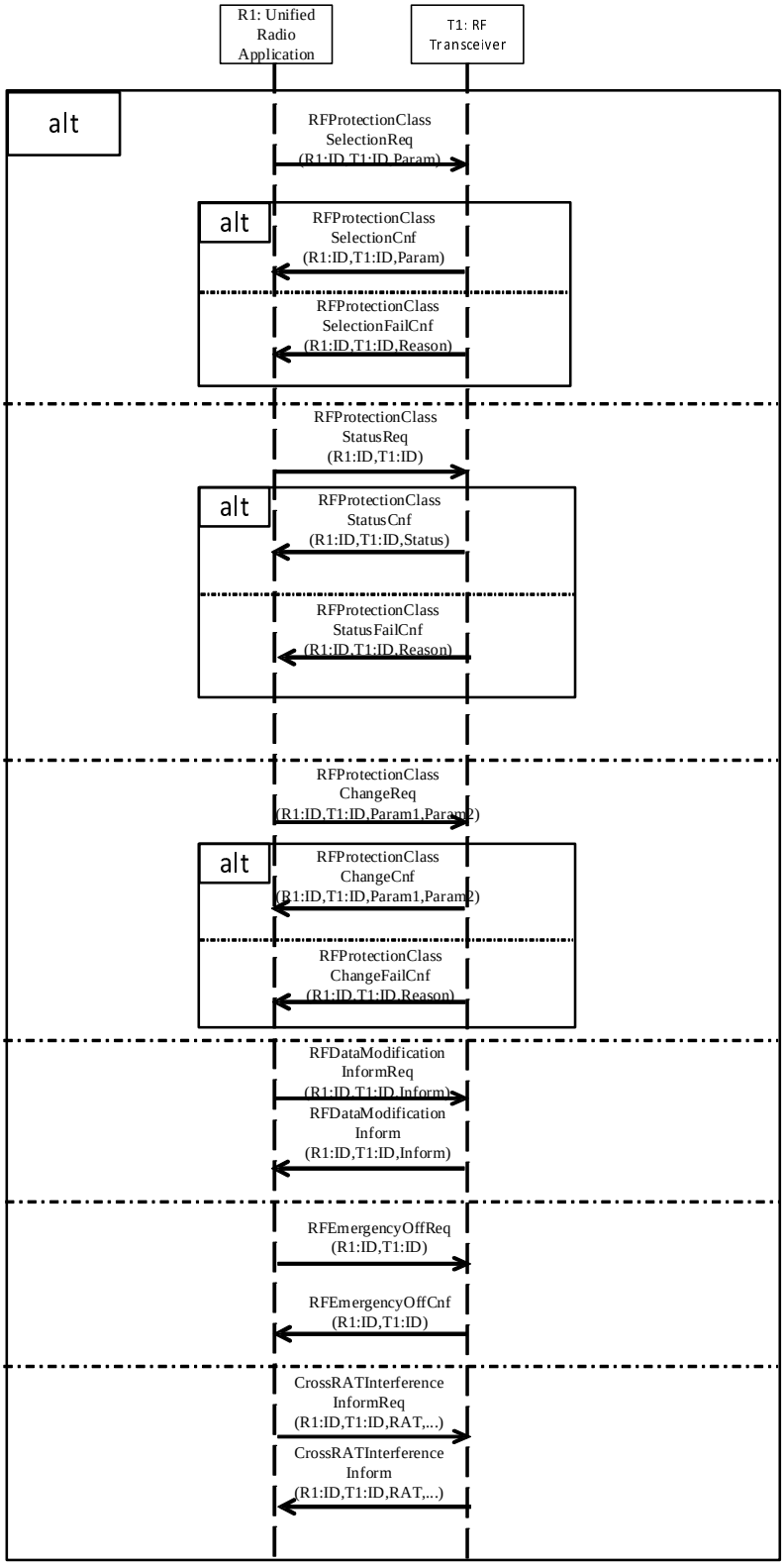

Figure 6.20: RVM protection services

The procedure can be summarized as follows:

- In the case of request of RF protection class selection, URA transfers a RFProtectionClassSelectionReq signal including URA (R1) ID, RF Transceiver (T1) ID, and RF protection class parameters to RF transceiver.
  - Upon completion of request of RF protection class selection, URA transfers a RFProtectionClassSelectionCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and RF protection class parameters to RF transceiver.
  - In the case of request of RF protection class selection, URA transfers a RFProtectionClassSelectionFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.
- In the case of request of RF protection class status, URA transfers a RFProtectionClassStatusReq signal including URA (R1) ID and RF Transceiver (T1) ID to RF transceiver.
  - Upon completion of request of RF protection class status, URA transfers a RFProtectionClassStatusCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and RF protection status to RF transceiver.
  - In the case of request of RF protection class status, URA transfers a RFProtectionClassStatusFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.
- In the case of request of RF protection class change, URA transfers a RFProtectionClassChangeReq signal including URA (R1) ID, RF Transceiver (T1) ID, RF protection class parameter1, and RF protection class parameter2 to RF transceiver.
  - Upon completion of request of RF protection class change, URA transfers a RFProtectionClassChangeCnf signal including URA (R1) ID, RF Transceiver (T1) ID, RF protection class parameter1, and RF protection class parameter2 to RF transceiver.
  - In the case of request of RF protection class change, URA transfers a RFProtectionClassChangeFailCnf signal including URA (R1) ID, RF Transceiver (T1) ID, and failure reason to RF transceiver.
- In the case of request of RF data modification information, URA transfers a RFDataModificationInformReq signal including URA (R1) ID, RF Transceiver (T1) ID, RF data modification information to RF transceiver. Then, RF transceiver transfers a RFDataModificationInform signal including URA (R1) ID, RF Transceiver (T1) ID and RF data modification information to URA.
  - In the case of request of RF emergency off, URA transfers a RFEmergencyOffReq signal including URA (R1) ID and RF Transceiver (T1) ID to RF transceiver. Then, RF transceiver transfers a RFEmergencyOffReq signal including URA (R1) ID and RF Transceiver (T1) ID to URA.
  - In the case of request of cross RAT interface information, URA transfers a CrossRATInterfaceInformationReq signal including URA (R1) ID, RF Transceiver (T1) ID, and RAT to RF transceiver. Then, RF transceiver transfers a CrossRATInterfaceInform signal including URA (R1) ID, RF Transceiver (T1) ID, and RAT to URA.

6.12 Procedure for RE Configuration Policy endorsement, distribution, and validation

Figure 6.21 illustrates the procedure for provisioning the RE Configuration Policy to the Reconfigurable MD.

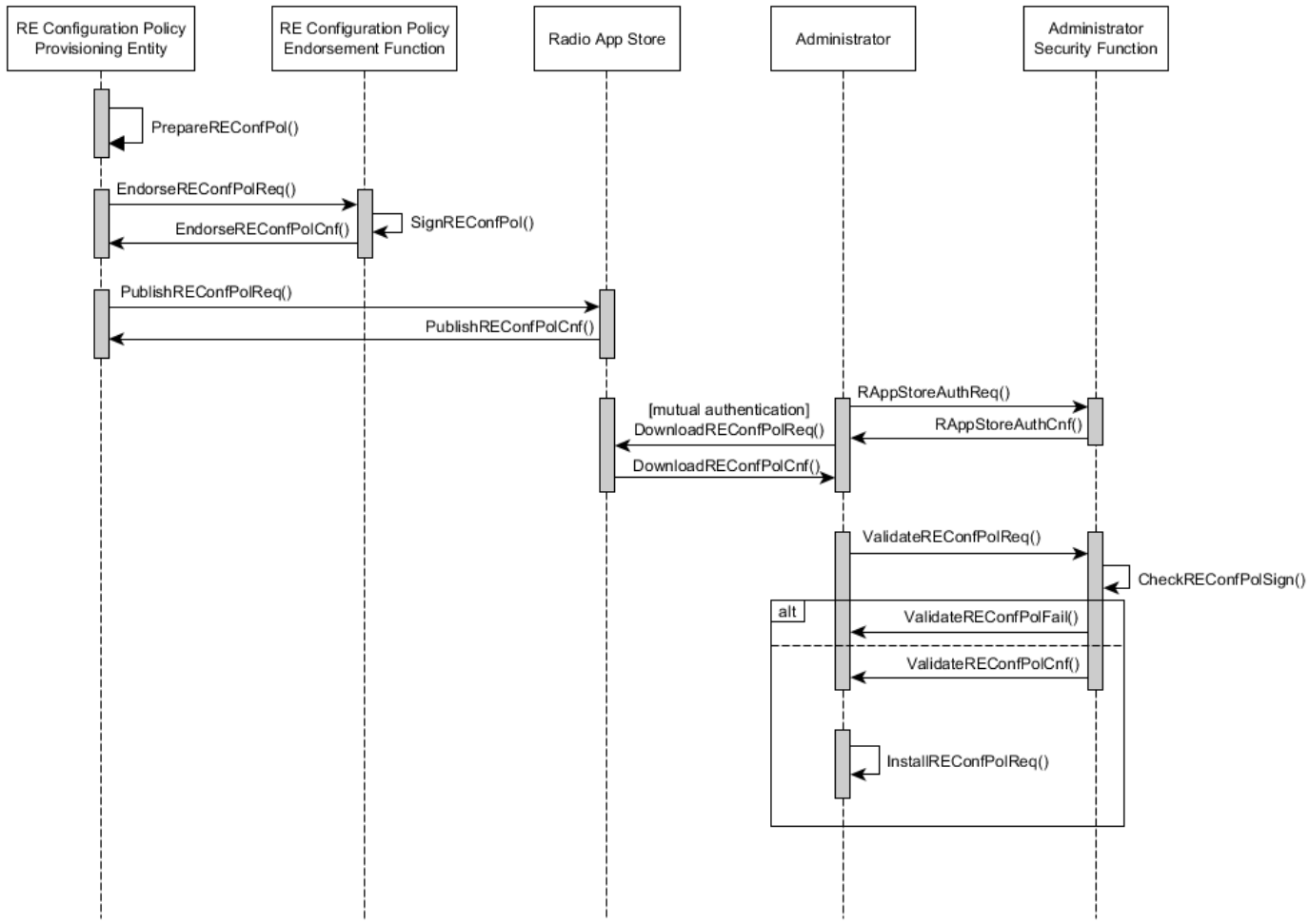

Figure 6.21: Signalling diagram for RE Configuration Policy endorsement, distribution and validation

The procedure can be summarized as follows:

- The RE Configuration Policy is assembled by the RE Configuration Policy Provisioning Entity (e.g. the device manufacturer).
- The RE Configuration Policy is submitted by the RE Configuration Policy Provisioning Entity to the RE Configuration Policy Endorsement Function (a specialization of the Asset Endorsement Functions) via the *EndorseREConfPolReq()* signal.
- The RE Configuration Policy Endorsement Functions signs the RE Configuration Policy via the reflexive *SignREConfPol()* signal.
- The RE Configuration Policy Provisioning Entity receives a *EndorseREConfPolCnf()* signal confirming successful endorsement (signature) the RE Configuration Policy.
- The RE Configuration Policy Provisioning Entity sends a *PublishREConfPolReq()* signal to the RadioApp Store for publication of the RE Configuration Policy.
- The RadioApp Store sends a *PublishREConfPolCnf()* signal to the RE Configuration Policy Provisioning Entity, confirming that the RE Configuration Policy is published and available for download by Reconfigurable MDs.
- The RadioApp Store and the Administrator on the Reconfigurable MD mutually authenticate each other. On the Reconfigurable MD this is done through an exchange of *RAppStoreAuthReq()* and *RAppStoreAuthCnf()* signals between the Administrator and the Administrator Security Function.
- The Administrator sends a *DownloadREConfPolReq()* signal to the RadioApp Store in order to download the RE Configuration Policy.
- The RadioApp Store sends a *DownloadREConfPolCnf()* signal to the Administrator.
- The Administrator sends a *ValidateREConfPolReq()* signal to the Administrator Security Function in order to verify the digital signature of the RE Configuration Policy. The verification operation is illustrated by the reflexive *CheckREConfPolSign()* signal.
    - Upon failure the Administrator Security Function sends a *ValidateREConfPolFail()* signal to the Administrator and the procedure is aborted.
    - Upon success the Administrator Security Function sends a *ValidateREConfPolCnf()* signal to the Administrator.
- The Administrator sends a reflexive *InstalREConfPolReq()* signal in order to install the RE Configuration Policy.

6.13 Procedure for configuration enforcement

Configuration enforcement is introduced in [i.5], clause 10.

Figure 6.22 illustrates the procedure for sending a remote control command by a remote controller.

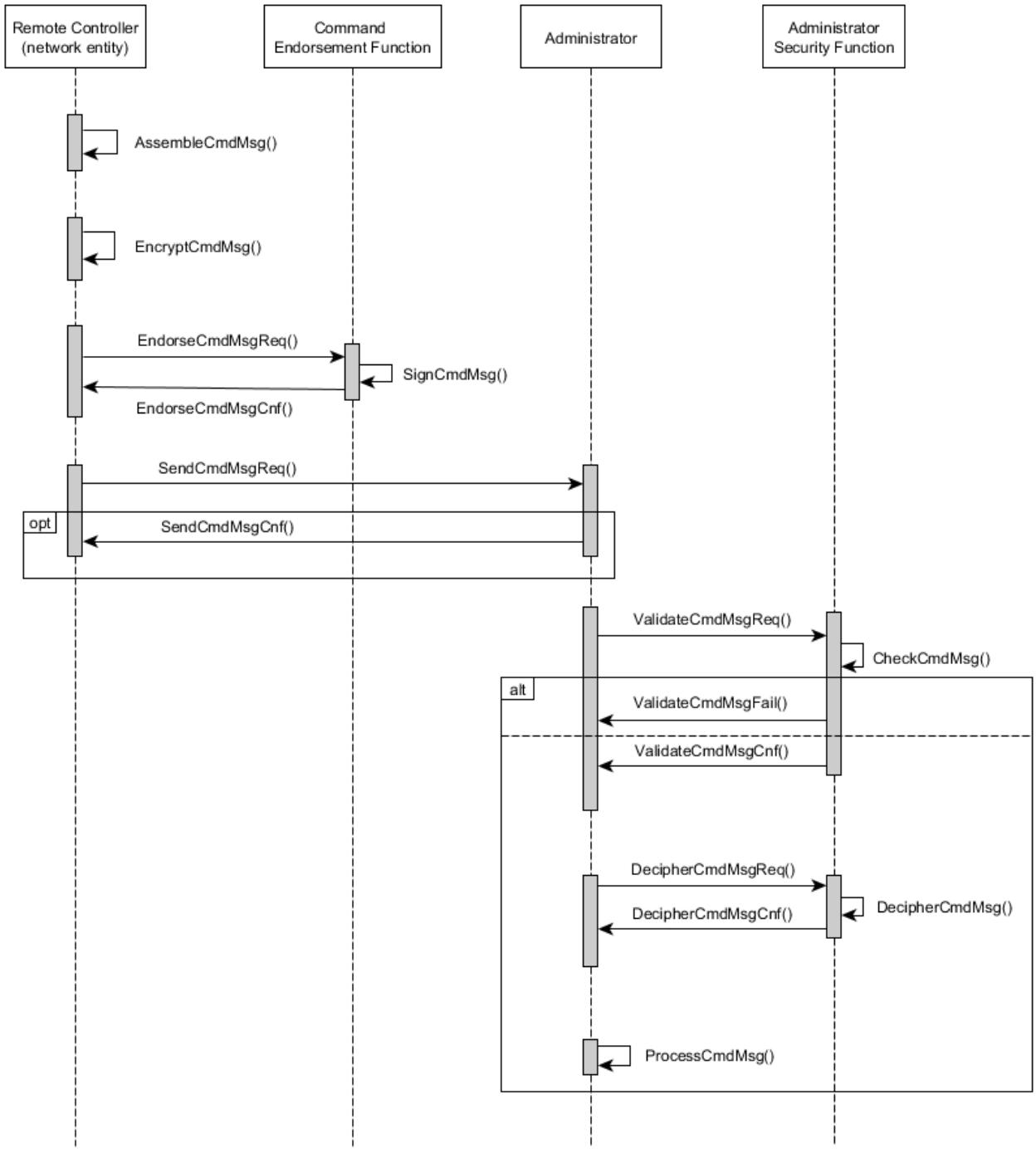

Figure 6.22: Signalling diagram for sending a remote control command

The procedure can be summarized as follows:

- The Remote Controller, a network entity that represents the sending party, first assembles a command message by sending a reflexive *AssembleCmdMsg()* signal.
- The Remote Controller encrypts the command message by sending a reflexive *EncryptCmdMsg()* signal.
- The encrypted command message is submitted by the Remote Controler to the Command Endorsement Function via an *EndorseCmdMsgReq()* signal.
- The Command Endorsement Function signs the encrypted command message via a reflexive *SignCmdMsg ()* signal.
- The encrypted and signed command message is returned from the Command Endorsement Function to the Remote Controller via an *EndorseCmdMsgCnf()* signal.

NOTE: The procedure explicitly follows the best practice of separating the encryption and digital signature operations - in particular with regard to key material. It would also be valid to have the Command Endorsement Function perform these two operations sequentially if using different keys, an option which is not illustrated in the diagram above.

- The Remote Controller sends the encrypted and signed command message to the Administrator on the Reconfigurable MD via a *SendCmdMsgReq()* signal.
  - If the message distribution system provides a return channel, the Administrator may confirm reception of the command message via a *SendCmdMsgCnf()* signal.
- The Administrator sends a *ValidateCmdMsgReq()* signal to the Administrator Security Function in order to verify the digital signature of the command message. The verification operation is illustrated by the reflexive *CheckCmdMsg()* signal.
  - Upon failure the Administrator Security Function sends a *ValidateCmdMsgFail()* signal to the Administrator and the procedure is aborted.
  - Upon success the Administrator Security Function sends a *ValidateCmdMsgCnf()* signal to the Administrator.
- The Administrator request decryption of the command message by sending a *DecipherCmdMsgReq()* signal to the Administrator Security Function. The decryption operation is illustrated by the reflexive *DecipherCmdMsg()* signal.
- The Administrator Security Function returns the deciphered and previously validated command message to the Administrator via a *DecipherCmdMsgCnf()* signal.
- The Administrator processes the command message. This is illustrated by the reflexive *ProcessCmdMsg()* signal.

6.14 Procedures for long-term management

Long-term management is introduced in [i.5], clause 11.

Figure 6.23 illustrates the transfer of authority between two RRS Configuration Authorities.

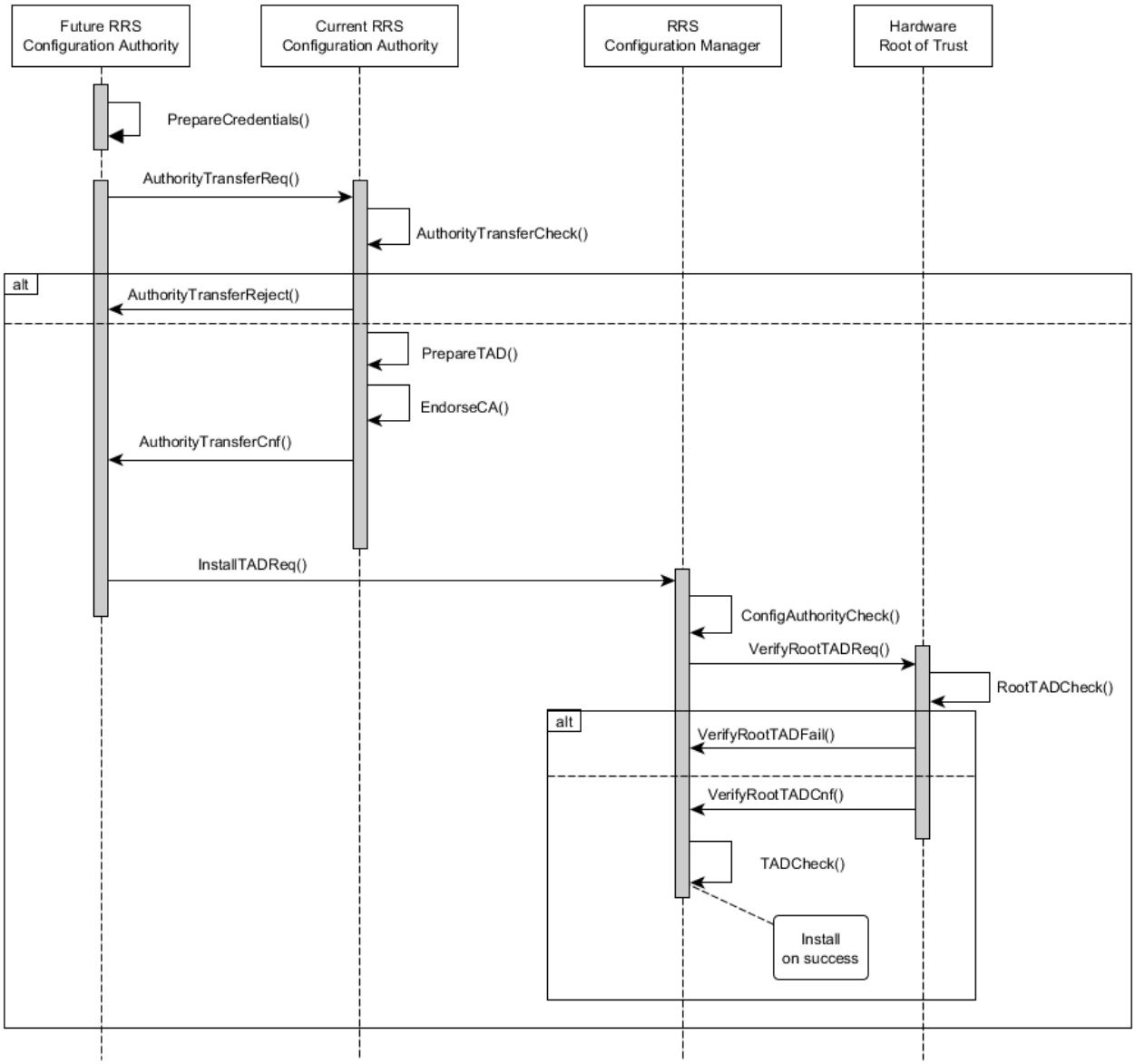

Figure 6.23: Signalling diagram for transferring authority between two RRS Configuration Authorities

The procedure can be summarized as follows:

- The future RRS Configuration Authority prepares its credentials via the reflexive *PrepareCredentials()* signal and requests a transfer of authority from the current RRS Configuration Authority via a *AuthorityTransferReq()* signal.

- The current RRS Configuration Authority verifies the validity of the transfer request via a reflexive *AuthorityTransferCheck()* signal.

  - Upon failure, the current RRS Configuration Authority sends a *AuthorityTransferReject()* signal and the procedure is aborted.

  - Upon success, the current RRS Configuration Authority prepares a Transfer of Authority Document (TAD) to the benefit of the future RRS Configuration Authority via the reflexive *PrepareTAD()* signal, and signs it via the reflexive *EndorseCA()* signal. The signed TAD is returned to the future RRS Configuration Authority via the *AuthorityTransferCnf()* signal.

NOTE 1: The steps described above should be viewed as a combination of administrative procedures between the two RRS Configuration Authorities, complemented by an electronic exchange protocol.

- The future RRS Configuration Authority indicates the transfer of authority to the Reconfigurable MD via an *InstallTADReq()* signal sent to the RRS Configuration Manager, providing the latter with the new, signed TAD.

- The RRS Configuration Manager verifies that the entity which signed the new TAD is the current RRS Configuration Authority via the reflexive *ConfigAuthorityCheck()* signal. This includes the verification of the complete chain of TADs.

- The RRS Configuration Manager sends a *VerifyRootTADReq()* signal to request verification of the root TAD by the Hardware Root of Trust.

- The Hardware Root of Trust verifies the root TAD via a reflexive *RootTADCheck()* signal.

  - Upon failure, the Hardware Root of Trust sends a *VerifyRootTADFail()* signal and the procedure is aborted.

  - Upon success, the Hardware Root of Trust sends a *VerifyRootTADCnf()* signal.

- The RRS Configuration Manager verifies the signature of the new TAD via the reflexive *TADCheck()* signal. Upon success the new TAD is installed, and the future RRS Configuration Authority, which so far was only a candidate from the point of view of the RRS Configuration Manager, effectively becomes the current RRS Configuration Authority.

Figure 6.24 illustrates the procedure for designating a legitimate RRS Configuration Provider RRS Configuration Authority.

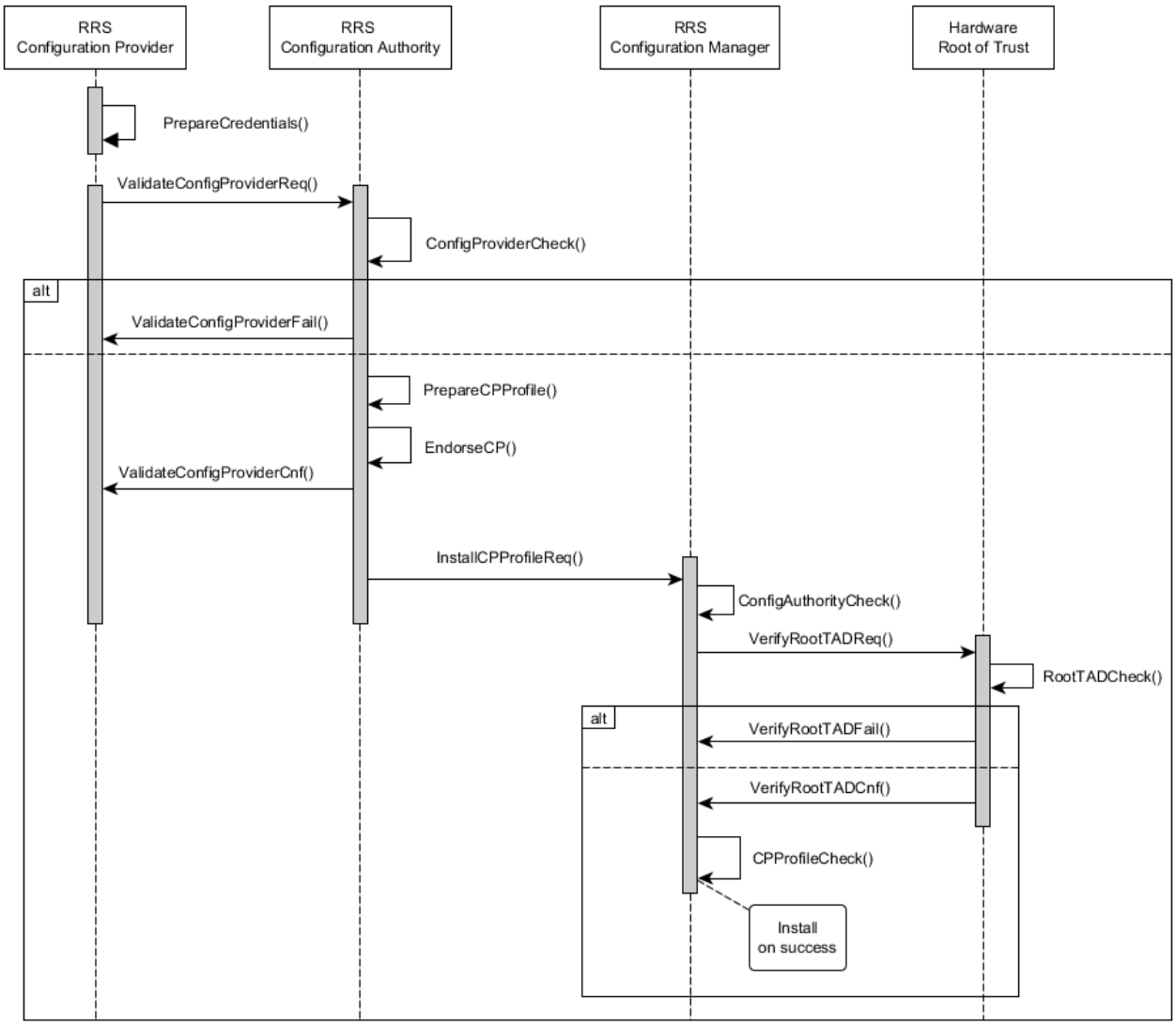

Figure 6.24: Signalling diagram for designating a legitimate RRS Configuration Provider

The procedure can be summarized as follows:

- The candidate RRS Configuration Provider prepares its credentials via the reflexive *PrepareCredentials()* signal and requests validation of its status as a configuration provider from the current RRS Configuration Authority via a *ValidateConfigProviderReq()* signal.

- The current RRS Configuration Authority verifies the validity of the request via a reflexive *ConfigProviderCheck()* signal.

  - Upon failure, the current RRS Configuration Authority sends an *ValidateConfigProviderFail()* signal and the procedure is aborted.

  - Upon success, the current RRS Configuration Authority prepares a RRS Configuration Provider Profile (or completes the one provide by the RRS Configuration Provider) via the reflexive *PrepareCPProfile()* signal, and signs it via the reflexive *EndorseCP()* signal. The signed profile is returned to the RRS Configuration Provider via the *ValidateConfigProviderCnf()* signal.

NOTE 2: The steps described above should be viewed as a combination of administrative procedures between the RRS Configuration Provider and the RRS Configuration Authority, complemented by an electronic exchange protocol.

- The current RRS Configuration Authority indicates the new RRS Configuration Provider to the Reconfigurable Equipement via an *InstallCPProfileReq()* signal sent to the RRS Configuration Manager, which provides the signed RRS Configuration Provider Profile.

- The RRS Configuration Manager verifies that the entity which signed the received RRS Configuration Profile is the current RRS Configuration Authority via the reflexive *ConfigAuthorityCheck()* signal. This includes the verification of the complete chain of TADs.

- The RRS Configuration Manager sends a *VerifyRootTADReq()* signal to request verification of the root TAD by the Hardware Root of Trust.

- The Hardware Root of Trust verifies the root TAD via a reflexive *RootTADCheck()* signal.

  - Upon failure, the Hardware Root of Trust sends a *VerifyRootTADFail()* signal and the procedure is aborted.

  - Upon success, the Hardware Root of Trust sends a *VerifyRootTADCnf()* signal.

- The RRS Configuration Manager verifies the signature of the received RRS Configuration Provider Profile via the reflexive *CPProfileCheck()* signal. Upon success the RRS Configuration Provider Profile is installed, and the candidate RRS Configuration Provider is accepted by the Reconfigurable MD.

Figure 6.25 illustrates the distribution of an RRS Configuration Profile by the RRS Configuration Providers.

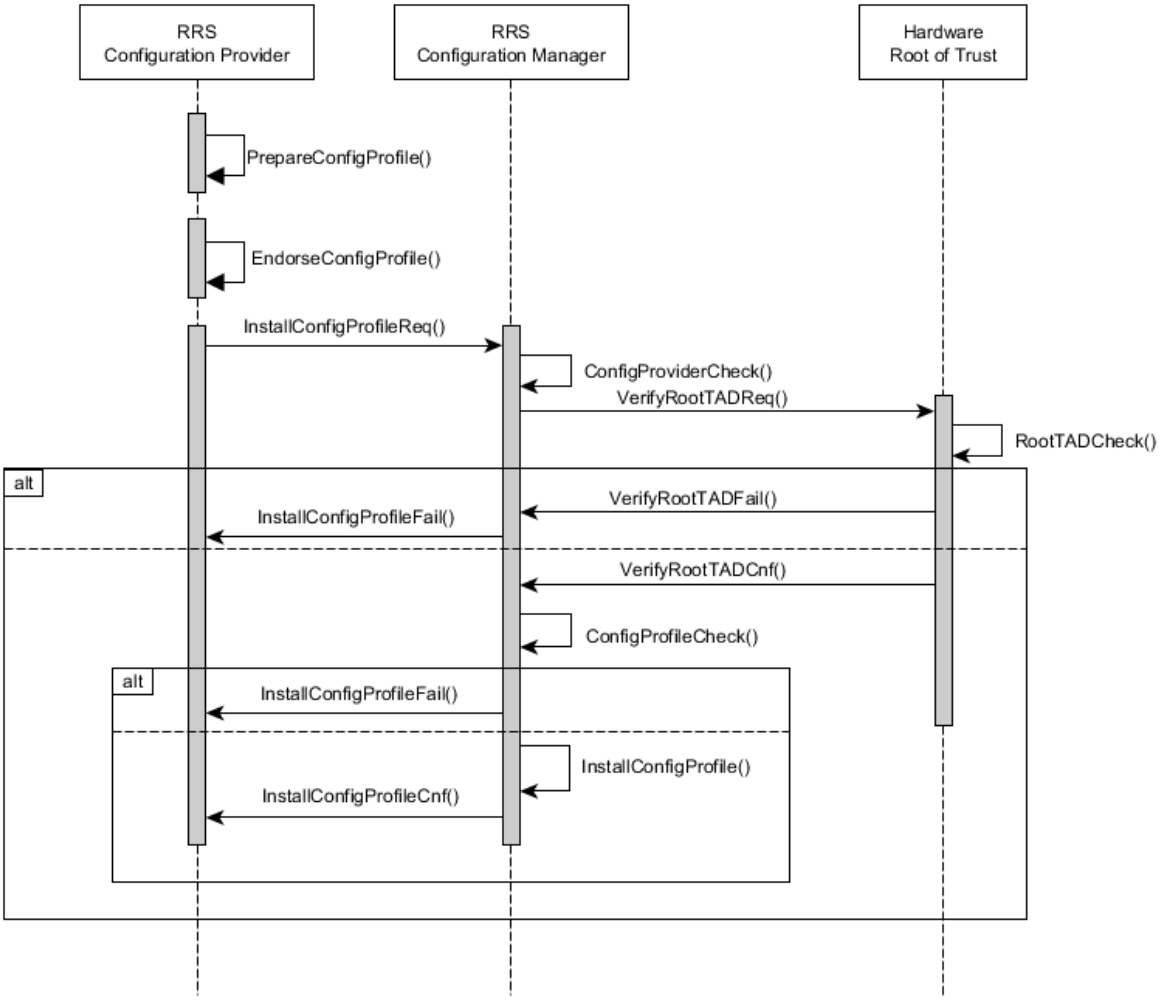

Figure 6.25: Signalling diagram for distributing RRS Configuration Profiles

The procedure can be summarized as follows:

- The RRS Configuration Provider prepares a new RRS Configuration Profile for the Reconfigurable MD via a reflexive *PrepareConfigProfile()* signal.
- The RRS Configuration Provider signs the new RRS Configuration Profile via a reflexive *EndorseConfigProfile()* signal.
- The RRS Configuration Provider sends an *InstallConfigProfileReq()* signal to the RRS Configuration Manager, in order to request installation of the new RRS configuration Profile on the Reconfigurable MD.
- The RRS Configuration Manager verifies that the RRS Configuration Provider is entitled to provide RRS Configuration Profiles to the Reconfigurable MD, via a reflexive *ConfigProviderCheck()* signal. This includes verification of the RRS Configuration Provider Profile and of the complete chain of TADs.
- The RRS Configuration Manager requests verification of the root TAD by the Hardware Root of Trust via a *VerifyRootTADReq()* signal.
- The Hardware Root of Trust verifies the root TAD via a reflexive *RootTADCheck()* signal.
    - Upon failure, the Hardware Root of Trust sends a *VerifyRootTADFail()* signal and the procedure is aborted. The RRS Configuration Manager may send an *InstallConfigProfileFail()* signal to the RRS Configuration Provider.
    - Upon success, the Hardware Root of Trust sends a *VerifyRootTADCnf()* signal.
- The RRS Configuration Manager verifies the signature of the new RRS Configuration Profile via a reflexive *ConfigProfileCheck()* signal.
    - Upon failure, the RRS Configuration Manager sends an *InstallConfigProfileFail()* signal to the RRS Configuration Provider.
    - Upon success, the RRS Configuration Manager installs the new RRS Configuration Profile, and sends an *InstallConfigProfileCnf()* signal to the RRS Configuration Provider.

History

Document history		
V1.1.1	January 2013	Publication as ETSI TS 103 095
V1.2.1	June 2015	Publication
V1.2.9	February 2018	EN Approval Procedure AP 20180517: 2018-02-16 to 2018-05-17
V1.3.1	May 2018	Publication