ETSI TS 103 095 V1.1.1 (2013-01)

Technical Specification

Reconfigurable Radio Systems (RRS);
Radio Reconfiguration related Architecture for Mobile Devices

Reference

DTS/RRS-02007

Keywords

architecture, mobile, SDR

Contents

Intellectual Property Rights

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Reconfigurable Radio Systems (RRS).

1 Scope

The scope of the present document is to define the radio reconfiguration related architecture for mobile devices. The work will be based on the Use Cases defined in TR 103 062 [i.1], TR 102 839 [i.2] and TR 102 944 [i.3] and on the system requirements defined in TS 102 969 [1].

2 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 http://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.

2.1 Normative references

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

[1] ETSI TS 102 969 (V1.1.1): "Reconfigurable Radio Systems (RRS); Radio Reconfiguration related Requirements for Mobile Devices".

2.2 Informative references

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 102 839: "Reconfigurable Radio Systems (RRS); Multiradio Interface for Software Defined Radio (SDR) Mobile Device Architecture and Services".

[i.3] ETSI TR 102 944: "Reconfigurable Radio Systems (RRS); Use Cases for Baseband Interfaces for Unified Radio Applications of Mobile Device".

[i.4] Joseph Mitola III (2000): "Software Radio Architecture", JOHN WILEY & SONS, INC.

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

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

NOTE: 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 source codes of 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 Intermediate Representation (IR) which should be back-end compiled for operating on a specific target platform.

Radio Application (RA): software which enforces RVM or particular radio platform to generate the transmit RF signals or decode the receive RF signals

NOTE: Radio Applications 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;
- IR including Radio Library calls of Radio Library native implementation and Radio HAL calls;
- executable codes for particular radio platform.

Radio Control Framework (RCF): control framework which, as a part of 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 Lib: library of Standard Function Block (SFB)

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 Radio Application developers.

NOTE 2: SFB is implemented through Radio HAL when the SFB is to be implemented on dedicated HW accelerators.

NOTE 3: 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 radio processor such as resource management, file system support, unified access to hardware resources, etc.

Radio Processor (RP): part of mobile device hardware working under ROS control and on which Radio Applications are executed

NOTE: RP is hardware which is capable to generate RF signals or receive 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. microprocessors, FPGAs, etc.

Radio Virtual Machine (RVM): abstract parallel machine capable to execute computations specific for radio signal processing

NOTE: RVM provides abstract vision of radio platform. To be efficient in different implementations it should represent specific features inherent to radio signal processing. Particularly RVM is capable to generate or receive RF signals.

3.2 Symbols

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

M_1	Number of SFBs implemented on core processor
M_2	Number of SFBs implemented on dedicated hardware logic accelerators

3.3 Abbreviations

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

AOT	Ahead-Of-Time
AP	Application Processor
ASIC	Applications-Specific Integrated Circuit
BBI	BaseBand Interface
BPA	Baseband Parameter Aggregation
CM	Configuration Manager
FC	Flow Controller
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FM	File Manager
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GPS	Global Positioning System
HAL	Hardware Abstraction Layer
ID	Identification
IP	Internet Protocol
IR	Intermediate Representation
ISA	Instruction Set Architecture
JIT	Just-In-Time
MD	Mobile Device
MIMO	Multi-Input-Multi-Output
MPM	Mobility Policy Manager
MRC	MultiRadio Controller
MURI	MUltiRadio Interface
OS	Operating System
RA	Radio Application
RAP	Radio Application Package
RC	Radio Controller
RCF	Radio Control Framework
RCM	Radio Connection Manager
RF	Radio Frequency
RM	Resource Manager
ROS	Radio Operating System
RP	Radio Processor
RRFI	Reconfigurable Radio Frequency Interface
RVM	Radio Virtual Machine
SDR	Software Defined Radio
SFB	Standard Function Block
UDFB	User Defined Function Block
URA	Unified Radio Applications
URAI	Unified Radio Applications Interface
WLAN	Wireless Local Area Network
XCVR	Transceiver

4 Overview of Radio Reconfiguration related Architecture Reference Model for Mobile Devices

4.1 Reconfigurable Mobile Device Architecture for Multiradio Applications

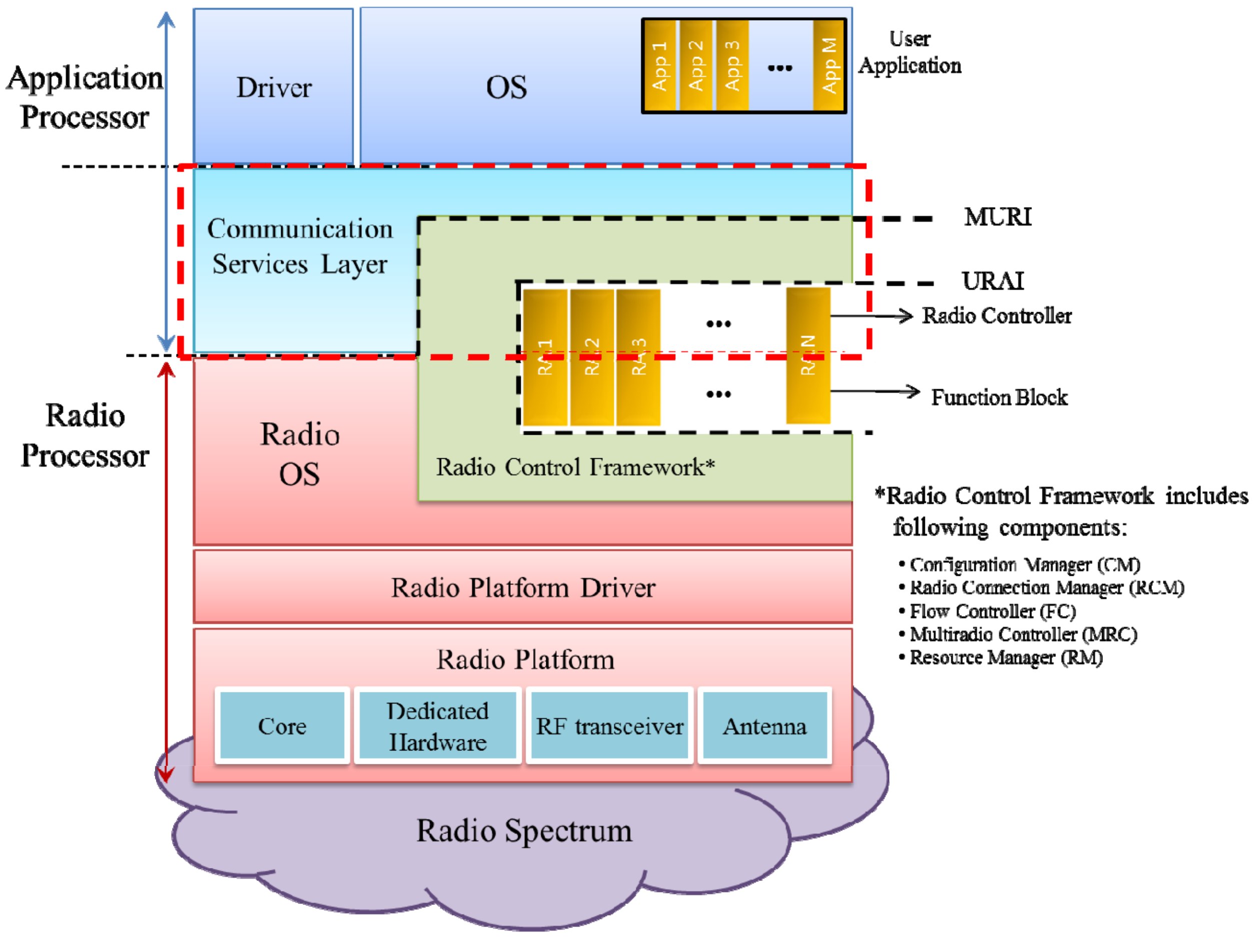

Figure 4.1: Reconfigurable Mobile Device (MD) architecture for multiradio applications

Figure 4.1 illustrates conceptual diagram of Reconfigurable MD architecture for multiradio applications. The Reconfigurable MD architecture shown in Figure 4.1 consists of Application Processor (AP) and Radio Processor (RP). Note that the red-dotted part might belong to RP instead of AP depending on each vendor.

Operation of AP is performed by a given Operating System (OS), which is in many cases performed on non-real-time bases, whereas RP's operation is performed by another OS, which should support real-time operations of RA. The OS of RP is referred to as ROS in the present document.

The AP includes the following components as shown in Figure 4.1.

- Driver activates hardware devices (such as camera, speaker, etc.) on a given MD.
- OS denotes a non-real-time operating system that operates in MD. As discussed in [i.3], the OS includes Administrator, Networking stack, Mobility Policy Manager (MPM) and Monitor. In clause 5 of the present document, Communication services layer as represented in [i.4] is introduced to represent the four entities of the OS [i.3]. In addition, for multiradio applications, the OS may include Radio Control Framework (RCF) (AP part) as well as the four entities [1].

- Radio Controller (RC) in Radio Application (RA) sends context information to the Monitor or sends/receives data to/from Networking stack.

The components of the RCF are classified into two groups. One group interfaces with the AP part and the other group with the RP part as shown in Figure 4.1. Which components of RCF interface to RP and which interface to AP, in real-time and in non-real-time, respectively, can be determined differently by each vendor.

- The RCF includes the following 5 components for managing RA(s) [i.2]:

  1) Configuration Manager (CM) provides for installation/uninstallation and creating/deleting instance of RAs into RP as well as management of and access to the radio parameters of the RAs.

  2) Radio Connection Manager (RCM) provides activation/deactivation of RAs according to user requests, and overall management of user data flows, which can also be switched from one RA to another.

  3) Flow Controller (FC) is responsible for sending and receiving of user data packets and controlling the flow of signalling packets.

  4) Multiradio Controller (MRC) schedules the requests for radio resources issued by concurrently executing RAs and detects and manages the interoperability problems among the concurrently executing RAs.

  5) Resource Manager (RM) manages the computational resources to share them among simultaneously active RAs, and to guarantee their real-time requirements.

The RP includes the following components as shown in Figure 4.1.

- ROS is a real-time OS that includes RCF (RP part).

- Radio platform driver is a hardware driver for the ROS to interact with the Radio platform hardware.

- Radio platform hardware typically consists of core(s) and baseband accelerator(s). Baseband accelerator implementing the Standard Function Block(s) (SFB) might be provided in a form of an Application-Specific Integrated Circuit (ASIC).

4.2 Software Architecture for Radio Processor

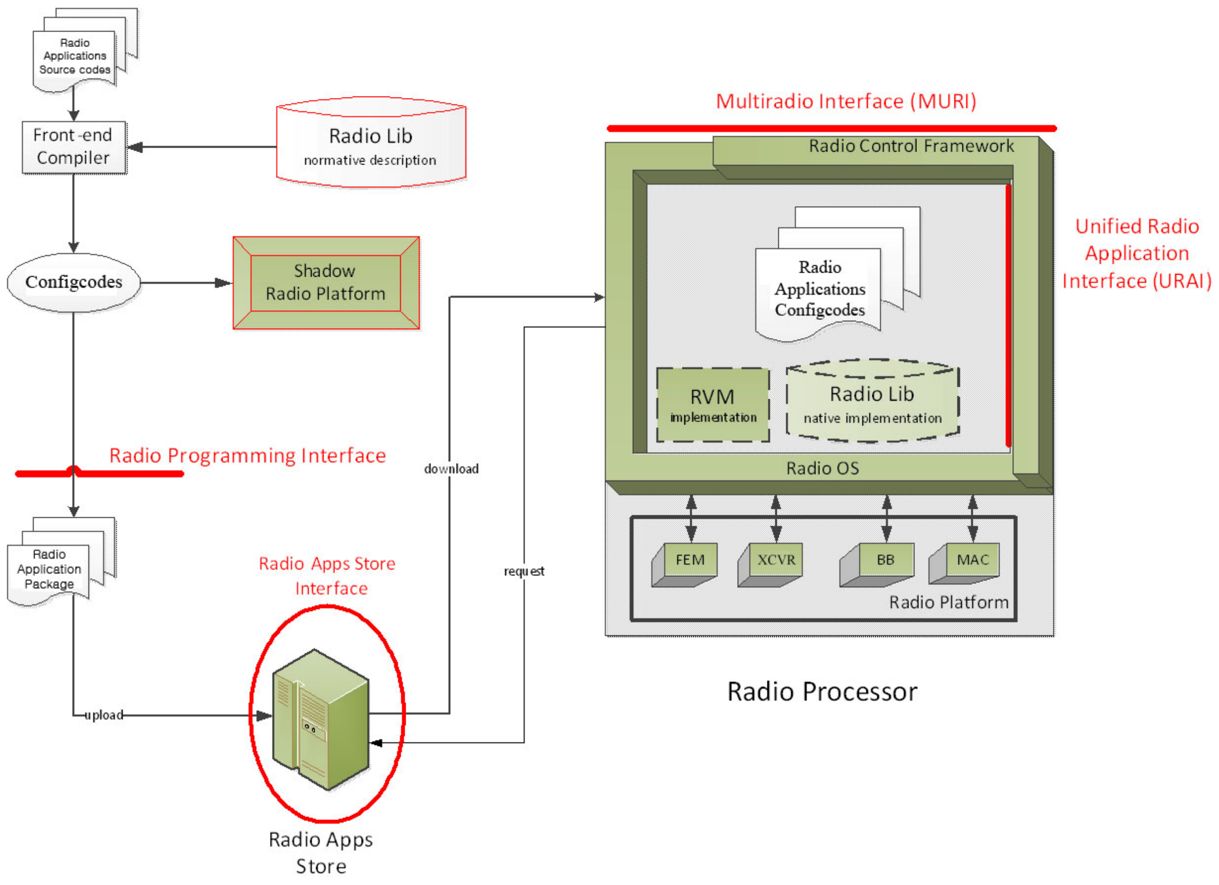
Radio Applications Source codes
Front-end Compiler
Radio Lib normative description
Configcodes
Shadow Radio Platform
Radio Programming Interface
Radio Application Package
Radio Apps Store Interface
upload
download
request
Radio Apps Store
Multiradio Interface (MURI)
Radio Control Framework
Unified Radio Application Interface (URAI)
Radio Applications Configcodes
RVM implementation
Radio Lib native implementation
Radio OS
FEM
XCVR
BB
MAC
Radio Platform
Radio Processor

Figure 4.2: Software architecture for RP

The software architecture of RP is illustrated in Figure 4.2. The RP provides communication capabilities for MDs and consists of:

- ROS including RCF.
- Radio Virtual Machine (RVM) implementation if Shadow radio platform is equal to RVM.
- Radio Lib native implementation if Shadow radio platform is equal to RVM.
- RAs configuration codes (configcodes).

Multiradio Interfaces (MURI) and Unified Radio Application Interfaces (URAI) are interfaces of RCF. As described in [i.2], MURI are interfaces between component of Communication services layer and that of RCF and URAI are interfaces between URA and component of RCF.

The Shadow radio platform can be either RVM or a target radio platform.

- If the Shadow radio platform is equal to the target radio platform, then front-end compiler will generate executable code for the target radio platform and configcodes is equivalent to the executable code for that radio platform.
- The RVM is Abstract Machine which is capable of executing configcodes. It is independent of the hardware. Implementation is a specific RVM implementation on the target radio platform. This implementation 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.

The Radio Lib consists of function blocks representing the computational basis. The RA can be expressed as a set of these interconnecting function blocks. Function blocks from the Radio Lib are represented in the normative language of the radio platform. Native implementation of Radio Lib provides executable codes of function blocks from the library for the target platform. Radio Lib is extendable.

RAs configcodes are interpreted by RVM when Shadow radio platform is equal to RVM, or are equal to executable codes when RVM is equal to target radio platform.

As illustrated in Figure 4.2, the access to a RadioApp Store is expected to require an interface. This is out of scope of the present document.

4.3 Operational Structure of URA

In this clause, operational structure of Unified Radio Applications (URA) is presented considering 2 different cases. One case is when RA configcodes are executable on a target radio platform and the other case is when RA configcodes are Intermediate Representation (IR) that is to be back-end compiled at a given MD.

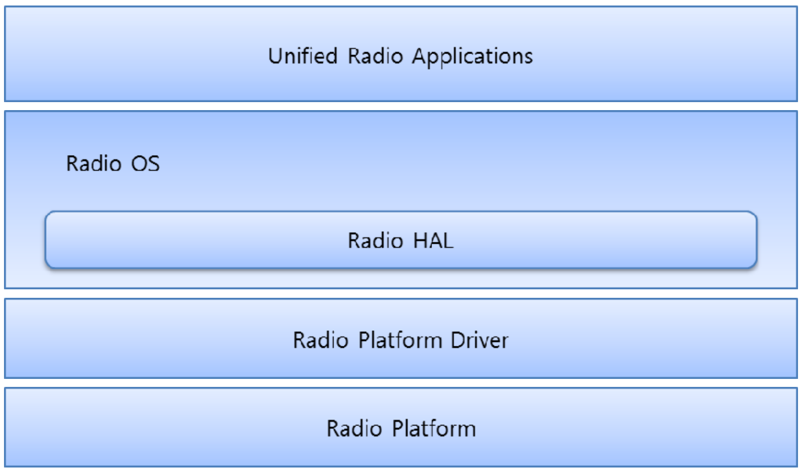

Figure 4.3: Operational structure of URA when RA configcodes are executable on a target platform

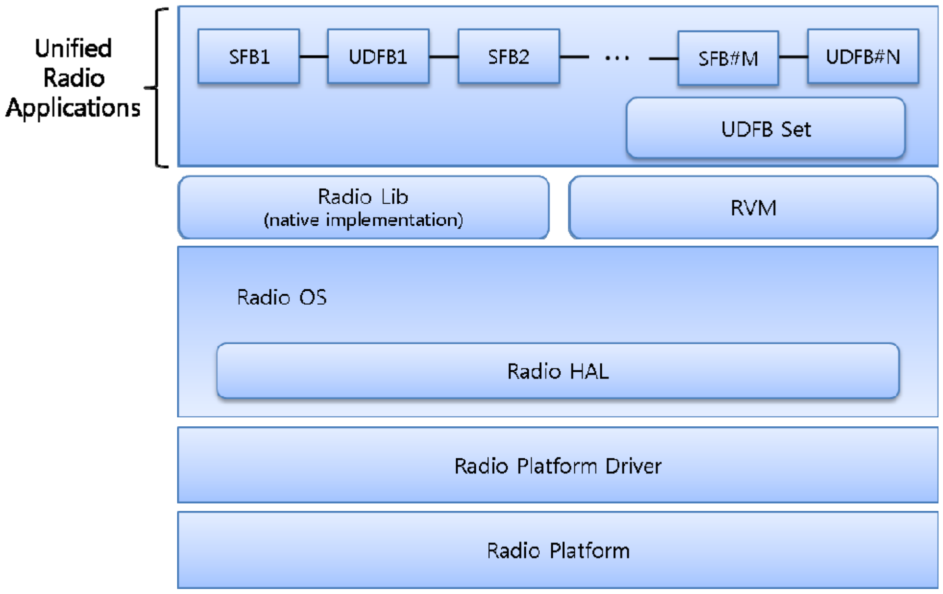

Figure 4.4: Operational structure of URA when RA configcodes are IR to be back-end compiled

Figure 4.3 illustrates an operational structure of URA when RA configcodes are executable on the target radio platform. Note that the Radio Lib and User Defined Function Blocks (UDFBs) needed to perform a given RA(s) are already bound in the executable configcodes of RA. Figure 4.4 illustrates an operational structure of URA when RA configcodes are IR that should be back-end compiled at a given MD. In this case, the UDFBs needed to perform a given RA(s) are included in the configcodes of RA and should be back-end compiled by RVM as shown in Figure 4.2. Note that the native implementation of Radio Lib should be prepared in a given MD separately because the Radio Lib cannot be contained in RA configcodes in this case. Generally, the native implementation of Radio Lib is provided by the core chip vendor because Radio Lib includes SFB(s) that is(are) implemented on core processor. Basically, Radio Lib (native implementation) shown in Figure 4.4 is needed for speed up of the SFBs that can be implemented without using dedicated hardware accelerator(s) or for combining accelerator(s) and program code to generate another SFB(s).

In either case when the RA configcodes are executable or IR, as shown in both Figures 4.3 and 4.4, Radio Hardware Abstraction Layer (HAL) includes hardware abstraction for SFB implementation using dedicated hardware logic accelerator(s). It particularly means that, whenever the SFB(s) to be implemented using dedicated hardware accelerator(s) is(are) called in a given RA code, it is implemented directly on a corresponding dedicated hardware logic accelerator(s) via the Radio HAL in either case when the RA configcodes are executable or IR.As will be discussed later in this clause, Radio HAL also includes hardware abstraction for interfaces prepared for UDFB Library(-ies).

Function blocks that are used in many RAs in common, e.g. Fast Fourier Transform (FFT), and/or some function blocks that should be implemented very efficiently using special-purpose accelerator(s) in a given radio platform, e.g. Turbo coder, are also to be defined as SFB(s).

Meanwhile, UDFB Set shown in Figure 4.4 includes all the UDFBs to be used in a given RA(s). It is important that any SFB can be modified and/or extended by replacing it with proper UDFB(s) which is a modified and/or extended version of the SFB to be replaced. Therefore, some UDFB(s) can be good candidate(s) for SFB extension, which means they might be added as SFB(s) later. In that case, after addition, they will become atomic as the normal SFBs. Since UDFB Set is to be provided by RA provider, i.e. 3rd party, instead of radio platform vendor, in order for RCF to be able to perform basic controls of every UDFB's event and/or command, a standard set of control interfaces such as "start", "*stop*", "*pause*", "*get_port*" and "*initialize*" may have to be specified for the corresponding UDFB(s). For this purpose, if necessary, a dedicated deliverable will specify a standard set of control interfaces for each UDFB to be implemented properly via the standard set of control interfaces. Specification of the standard control interfaces for UDFB(s) as well as SFB(s) will be given in the document of Protocol/Interface TS. Recall that radio platform shown in Figures 4.3 and 4.4 generally consists of both core(s) and dedicated hardware accelerator(s) for implementing each of function blocks.

Operational structure of URA includes the following components as shown in Figure 4.4.

- RA includes SFB(s) and UDFB(s) in accordance with the contents of metadata in a given Radio Application Package (RAP). Recall that Baseband Interface (BBI) represents each function block itself by specifying the name of the corresponding function block. It also specifies interface related to the corresponding function blocks as mentioned earlier [i.3].
- Radio Lib (normative implementation) contains configcodes of SFBs that are to be implemented on core processor(s) while the SFBs that are to be implemented using dedicated hardware logic accelerator(s) are supported by Radio HAL.
- UDFB Set includes all the UDFBs to be used in a given RAP and is in general provided by RA provider. UDFB is included in RAP together with metadata and RC code [i.3]. Since UDFB is in general modified and/or extended version of SFB, UDFB shall have a dependency on SFB library(-ies).
- Radio HAL is to abstract radio platform. Note that Radio HAL supports SFB to be implemented using dedicated hardware logic accelerator in order for each of those SFBs to be implemented directly on corresponding dedicated hardware logic accelerator(s).
- Radio platform driver is for ROS to recognize radio platform.
- Radio platform in general consists of both core(s) and dedicated hardware accelerator(s).

Figure 4.5 illustrates implementation of function blocks on a given radio platform which consists of core(s) and various kinds of peripheral devices. In the example shown in Figure 4.5, the number of SFBs implemented on core processor has been set to M_1 and the number of SFBs implemented on dedicated hardware logic accelerators has been set to M_2. As mentioned earlier in this clause, SFBs that are to be implemented using dedicated hardware logic accelerator such as, for example, FFT, Turbo decoder, Multi-Input-Multi-Output (MIMO) decoder, etc., can be implemented directly on the corresponding dedicated hardware logic accelerator for high performance and low power consumption. Those SFBs are supported by Radio HAL for implementation on the dedicated accelerator(s). It particularly means that, when each of SFBs to be implemented on the dedicated accelerator(s) is called in RA, it is implemented directly on the corresponding dedicated accelerator(s) through Radio HAL. Similarly, when each of SFBs to be implemented on core processor such as, for example, bit-reverse, multiply and accumulation, etc., is called in RA, it is implemented on a given core.

Consequently, the execution codes required on RP consists of the following two parts. One part is execution codes for SFBs implemented on programmable core(s) and the other part is Radio HAL codes for SFBs implemented on dedicated accelerators. It can be summarized as follows:

{C: exe codes required on RP for SFB implementation} = {A: exe codes for SFBs implemented on programmable cores} + {B: Radio HAL codes for SFBs implemented on accelerators},

meaning that:

$$C = A + B$$

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

{SFBs} = {SFBs implemented on core processor} $\cup$ {SFBs implemented on dedicated hardware accelerators}

where:

{SFBs implemented on core processor} $\cap$ {SFBs implemented on dedicated hardware accelerators} = $\emptyset$.

Meanwhile, UDFB, as mentioned earlier in this clause, should be written with standard interfaces. In Figure 4.5, it should be observed that the standard interfaces of UDFB might be associated with either SFB(s) implemented on core processor, or SFB(s) implemented on dedicated hardware accelerator, or both.

The reason why we classify standard interfaces into two groups, i.e. the one corresponding to SFB(s) implemented on core processor and the other corresponding to SFB(s) implemented on dedicated hardware accelerator, is that each category has its own pros and cons. The latter, since it is implemented on dedicated hardware logic, is advantageous for power consumption, speed-up operation, and, probably, cost-effectiveness while the former, since it is implemented on microprocessor, 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 more and more, the core-dependent SFB will gradually become more and more dominant compared to the core-and-peripheral-dependent SFB in a long term standpoint and be implemented via Instruction Set Architecture (ISA)-level acceleration.

Note that SFBs taken as examples in this clause are just for the purpose of explanation.

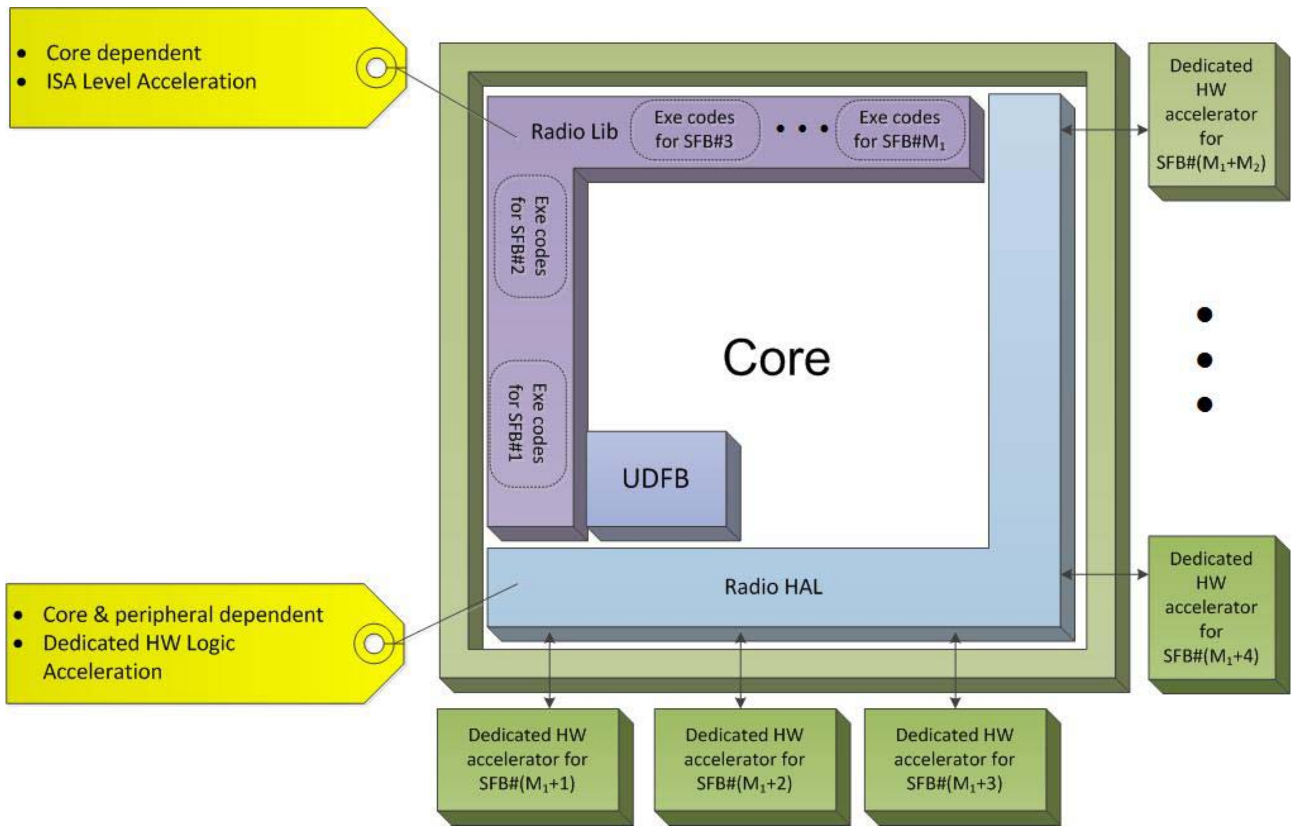
Core dependent
ISA Level Acceleration
Core & peripheral dependent
Dedicated HW Logic Acceleration
Radio Lib
Exe codes for SFB#3
Exe codes for SFB#M1
Exe codes for SFB#2
Exe codes for SFB#1
Core
UDFB
Radio HAL
Dedicated HW accelerator for SFB#(M1+M2)
Dedicated HW accelerator for SFB#(M1+4)
Dedicated HW accelerator for SFB#(M1+1)
Dedicated HW accelerator for SFB#(M1+2)
Dedicated HW accelerator for SFB#(M1+3)

Figure 4.5: Implementation of function block libraries on radio platform

5 Reference Points

Figure 5.1 illustrates entire architecture of MD with all the reference points being specified in between corresponding components. In Figure 5.1, each solid line between two blocks denotes a reference point 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 will be shown in this clause, 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 described in [i.2], i.e. MURI which are interfaces between component of Communication services layer and that of RCF, URAI which are interfaces between URA and component 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 components of RCF have also been defined as reference points. In the present document, we classify the reference points 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.

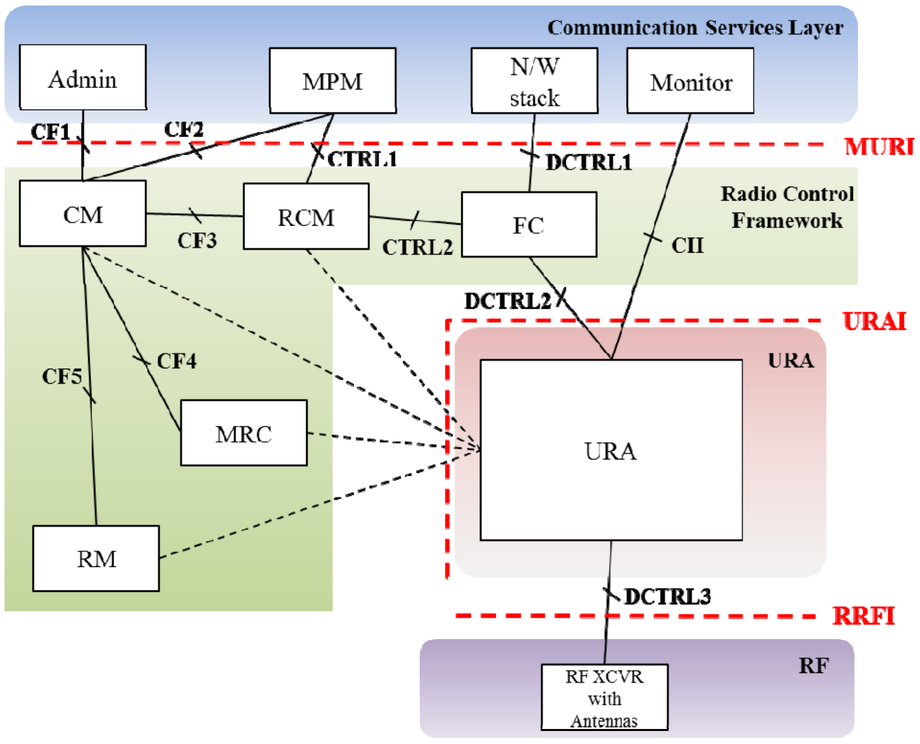

Figure 5.1: Entire architecture of reference points for MD

5.1 Reference Point 1: Interfaces for installation/uninstallation and creating/deleting instance of RA

5.1.1 List of Reference Point 1

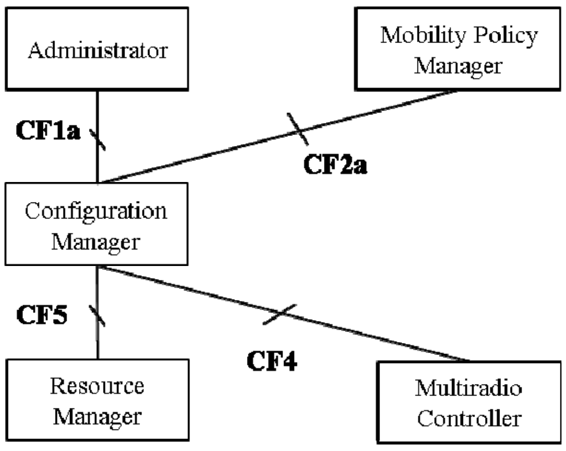

Figure 5.2: Illustration of reference points for installation/uninstallation and creating/deleting instance of RA

Figure 5.2 illustrates reference points, CF1a, CF2a, CF4, and CF5, of which the function is related to installation/uninstallation and/or creating/deleting instance of RA.

Reference Point **CF1a** is an interface between Administrator and CM, which is for Administrator to request CM to perform installing, uninstalling of RA or for Administrator to receive response of the request from CM.

Reference Point **CF2a** is an interface between MPM and CM, which is for MPM to request CM to perform creating instance or deleting instance of RA or for MPM to receive response of the request from CM.

Reference Point **CF4** is an interface between CM and MRC, which is for CM to request MRC to send parameters related to radio resources to CM, or for CM to receive response of the request, i.e. the parameters related to radio resources, from MRC during the procedure of creating instance of RA.

Reference Point **CF5** is an interface between CM and RM, which is for CM to request RM to send parameters related to computational resources to CM, or for CM to receive response of the request, i.e. the parameters related to computational resources, from RM during the procedure of creating instance of RA.

5.1.2 Reference Point 1 Requirements

- Administrator and CM should be interfaced in order for Administrator to request CM to perform installing, uninstalling of RA or for Administrator to receive response of the request from CM.
- MPM and CM should be interfaced in order for MPM to request CM to perform creating instance or deleting instance of RA or for MPM to receive response of the request from CM.
- CM and MRC should be interfaced in order for CM to request MRC to send parameters related to radio resources to CM, or for CM to receive response of the request, i.e. the parameters related to radio resources, from MRC during the procedure of creating instance of RA.
- CM and RM should be interfaced in order for CM to request RM to send parameters related to computational resources to CM, or for CM to receive response of the request, i.e. the parameters related to computational resources, from RM during the procedure of creating instance of RA.

5.2 Reference Point 2: Interfaces for list checking of RA(s)

5.2.1 List of Reference Point 2

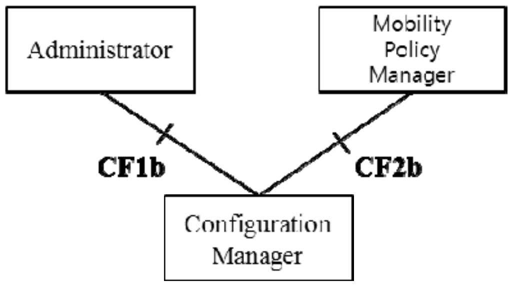

Figure 5.3: Illustration of reference points for obtaining the lists of RA(s)

Figure 5.3 illustrates reference points, CF1b and CF2b, of which the function is related to list checking of RA(s).

Reference Point **CF1b** is an interface between Administrator and CM, which is for Administrator to request CM to send the RA list to Administrator, or for Administrator to receive response of the request, i.e. the RA list, from CM.

Reference Point **CF2b** is an interface between MPM and CM, which is for MPM to request CM to send the RA list to MPM, or for MPM to receive response of the request, i.e. the RA list, from CM.

5.2.2 Reference Point 2 Requirements

- Administrator and CM should be interfaced in order for Administrator to request CM to send the RA list to Administrator, or for Administrator to receive response of the request, i.e. the RA list, from CM.
- MPM and CM should be interfaced in order for MPM to request CM to send the RA list to MPM, or for MPM to receive response of the request, i.e. the RA list, from CM.

5.3 Reference Point 3: Interfaces for activation/deactivation of RA

5.3.1 List of Reference Point 3

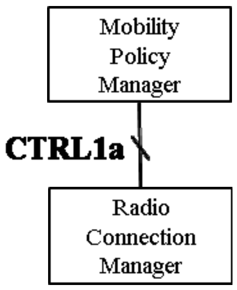

Figure 5.4: Illustration of reference point for activation/deactivation of RA

Figure 5.4 illustrates reference point, CTRL1a, of which the function is related to activation/deactivation of RA(s).

Reference Point **CTRL1a** is an interface between MPM and RCM, which is for MPM to request RCM to perform activation/deactivation of RA, or for MPM to receive response of the request, i.e. activation/deactivation of RA, from RCM.

5.3.2 Reference Point 3 Requirements

- MPM and RCM should be interfaced in order for MPM to request RCM to perform activation/deactivation of RA, or for MPM to receive response of the request, i.e. activation/deactivation of RA, from RCM.

5.4 Reference Point 4: Interfaces for transferring context information

5.4.1 List of Reference Point 4

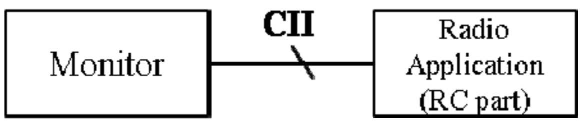

Figure 5.5: Illustration of reference points for transferring context information

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

Reference Point **CII** is an interface between Monitor and RC in RA, which is for Monitor to request RC in RA to send context information to Monitor, or for Monitor to receive response of the request, i.e. the context information, from RC in RA.

Explanation: The context information is generated from corresponding function block(s) of RA(s) and transferred to RC.

5.4.2 Reference Point 4 Requirements

- Monitor and RC in RA shall be interfaced in order for Monitor to request RC in RA to send context information to Monitor, or for Monitor to receive response of the request, i.e. the context information, from RC in RA.

5.5 Reference Point 5: Interfaces for creating data flow and sending/receiving user data

5.5.1 List of Reference Point 5

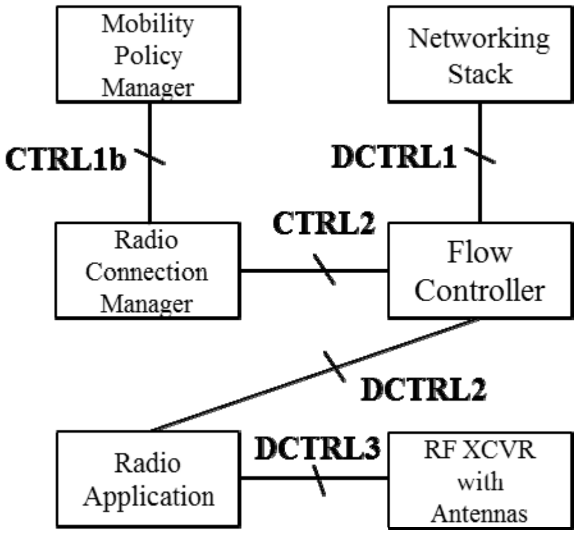

Figure 5.6: Illustration of reference points for creating data flow and sending/receiving user data

Figure 5.6 illustrates reference points, CTRL1b, CTRL2, DCTRL1, DCTRL2, and DCTRL3, of which the function is related to creating data flow and/or sending/receiving of user data.

Reference Point **CTRL1b** is an interface between MPM and RCM, which is for MPM to request RCM to form data flow or network association with peer equipment, or for MPM to receive response of the request from RCM.

Reference Point **CTRL2** is an interface between RCM and FC, which is for RCM to request FC to form data flow, or for RCM to receive response of the request from FC.

Reference Point **DCTRL1** is an interface between FC and Networking stack, which is for FC to receive/transfer user data from/to Networking stack for the procedure of sending/receiving data. It also includes an acknowledgement of transmit user data from FC to Networking stack upon completion of sending data.

Reference Point **DCTRL2** is an interface between FC and RA, which is for FC to transfer the transmit user data to RA and to request RA to transfer the information of transmit user data such as throughput, data bandwidth, etc to FC. DCTRL2 interface is also used for FC to receive response of the request from RA. In the case of the procedure of receiving data, DCTRL2 interface is used to transfer the receive user data from RA to FC.

Reference Point **DCTRL3** is an interface between RA and RF transceiver (XCVR) with antenna(s), which is for RA to receive/transfer receive/transmit user data from/to RF XCVR with antenna.

5.5.2 Reference Point 5 Requirements

- MPM and RCM should be interfaced in order for MPM to request RCM to form data flow or network association with peer equipment, or for MPM to receive response of the request from RCM.
- RCM and FC should be interfaced in order for RCM to request FC to form data flow, or for RCM to receive response of the request from FC.
- Networking stack and FC should be interfaced in order for FC to receive/transfer user data from/to Networking stack for the procedure of sending/receiving data. The interface between Networking stack and FC should also include an acknowledgement of transmit user data transfer from FC to Networking stack upon completion of sending data.

- FC and RA should be interfaced in order for FC to transfer the transmit user data to RA and to request RA to transfer the information of transmit user data such as throughput, data bandwidth, etc to FC. The interface between FC and RA should also include a procedure for FC to receive response of the request from RA. In the case of the procedure of receiving data, the interface between FC and RA should include a procedure of transferring the receive user data from RA to FC.
- RA and RF XCVR with antenna(s) shall be interfaced in order for RA to receive/transfer receive/transmit user data from/to RF XCVR with antenna.

6 Procedures

In this clause, procedures of MD operations performed with the reference points defined in the previous clause are introduced. Each of reference points defined in the previous clause is associated with each of corresponding procedures introduced in this clause.

6.1 Procedures for installation/uninstallation and creating/deleting instance of RA

In this clause, procedures related to installation/uninstallation and creating/deleting instance of RA are introduced. Figure 6.1 illustrates signalling diagram associated with the installation and uninstallation of RA.

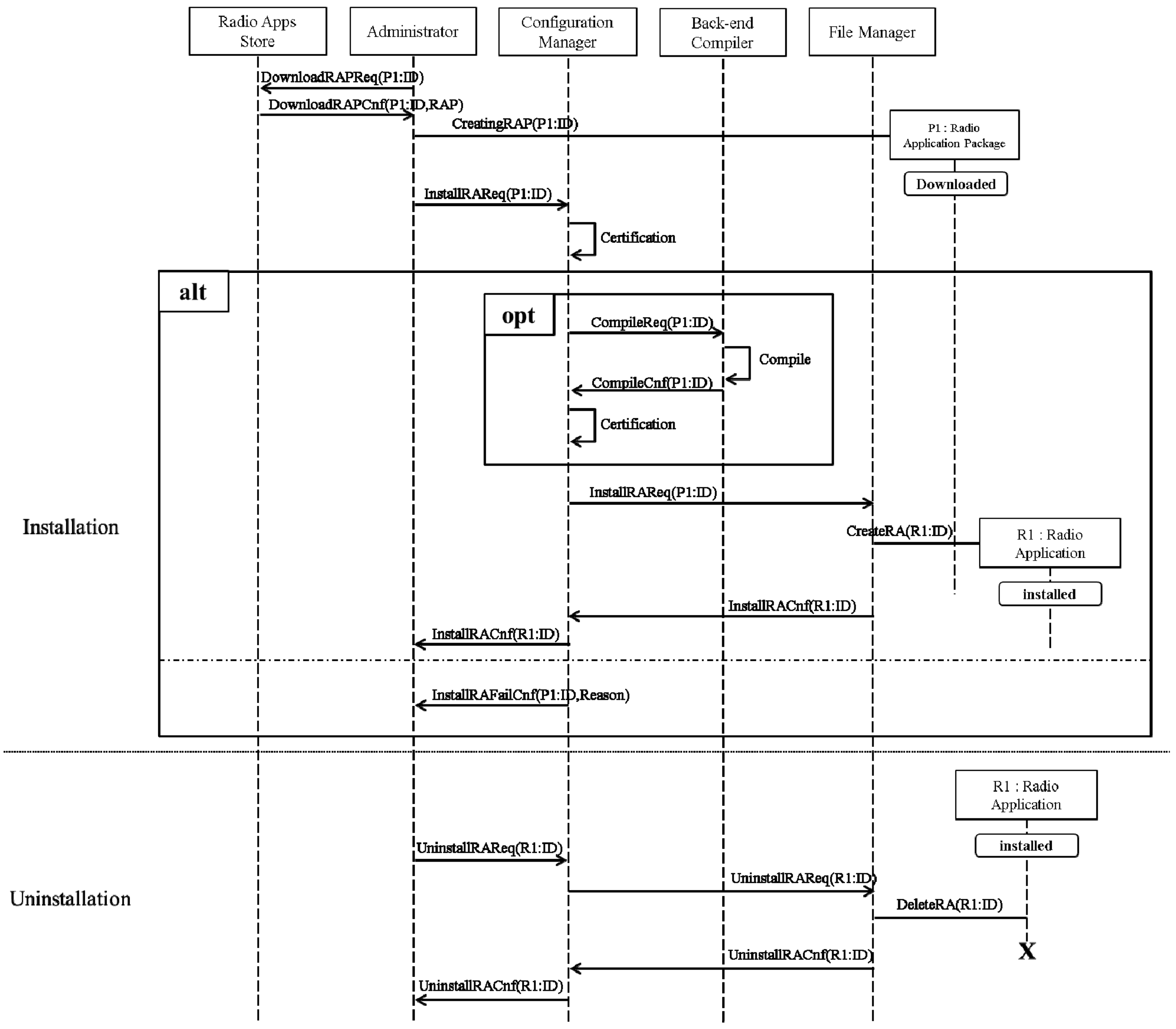

Figure 6.1: Signalling diagram of installation and uninstallation of RA

The procedure of installing RA shown in Figure 6.1 can be summarized as follows:

- Administrator sends *DownloadRAPReq* signal including RAP identification (ID) to Radio Apps Store.
- Administrator receives *DownloadRAPCnf* signal including RAP ID and RAP from Radio Apps Store.
- Administrator sends *InstallRAReq* signal including RAP ID to CM to request RA installation.
- CM first performs the procedure of certifying the RA code in order to verify its compatibility, authentication, etc.
- CM sends *InstallRAReq* signal including RAP ID to File Manager (FM) to perform installation of RA.
- FM performs installation of RA and transfers *InstallRACnf* signal including RA ID to CM, which transfers *InstallRACnf* signal including RA ID to Administrator.
- In the case when the downloaded RA is IR, CM first sends *CompileReq* signal including RAP ID to Back-end Compiler in advance of RA installation. After completion of back-end compilation, Back-end Compiler transfers *CompileCnf* signal including RAP ID to CM, which performs the procedure of certifying usability of the back-end compiled RA code.
- In the case of installation failure, CM reports Administrator the failure of RA installation using *InstallRAFailCnf* signal including RAP ID and failure reason.

The procedure of uninstalling RA shown in Figure 6.1 can be summarized as follows:

- Administrator transfers *UninstallRAReq* signal including ID of RA to be uninstalled to CM.
- CM sends *UninstallRAReq* signal including RA ID to FM.
- FM performs uninstallation of RA and transfers *UninstallRACnf* signal including RA ID as an acknowledgement of uninstallation completion to CM.
- CM sends *UninstallRACnf* signal including RA ID to Administrator.

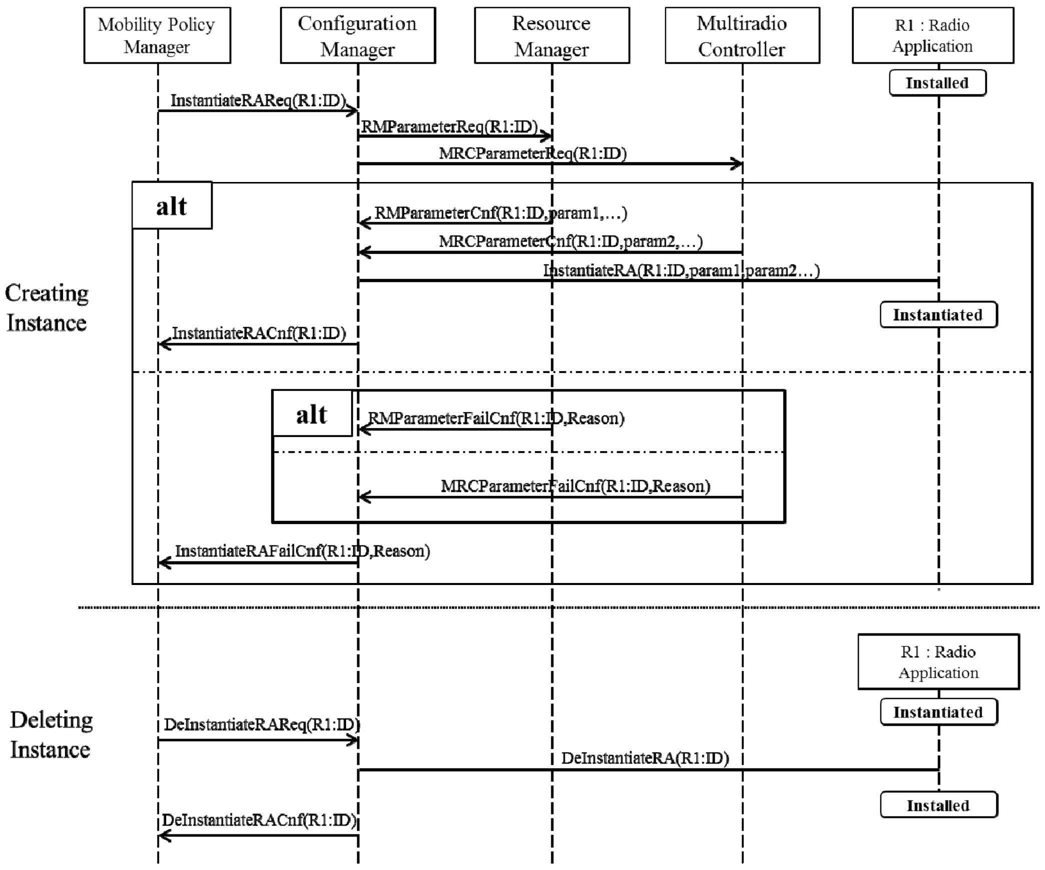

Figure 6.2: Signalling diagram of creating and deleting instance of RA

Figure 6.2 illustrates a signalling diagram showing the procedure of creating/deleting instance of RA.

The procedure of creating instance of RA shown in Figure 6.2 can be summarized as follows:

- For creating instance of installed RA, MPM transfers *InstantiateRAReq* signal including ID of RA to be instantiated to CM.
- CM transfers *RMParameterReq* signal and *MRCParameterReq* signal including ID of RA to get parameters needed for RA activation (e.g. Forward Error Correction (FEC) parameters, MIMO parameters, bandwidth, etc.) to RM and MRC.
- CM receives *RMParameterCnf* signal including ID of RA and radio resource parameters from RM.
- CM receives *MRCParameterCnf* signal including ID of RA and computational resource parameters from MRC.
- Upon completion of receiving parameters needed for RA activation, CM transfers RA ID and the parameters for performing the RA instantiation to ROS.
- Upon completion of creating instance, CM transfers *InstantiateRACnf* signal including RA ID to MPM.
- When CM fails to get parameters needed for RA activation from RM and/or MRC, RM and/or MRC reports the failure of parameters transfer to CM using *RMParameterFailCnf* and/or *MRCParameterFailCnf*, respectively.
- If the creating instance of RA fails, i.e. CM receives *RMParameterFailCnf* and/or *MRCParameterFailCnf* signal, CM reports the instantiation failure to MPM using *InstantiateRAFailCnf* signal.

The procedure of deleting instance of RA shown in Figure 6.2 can be summarized as follows:

- MPM transfers ID of RA to be deleted using *DeInstantiateRAReq* signal to CM.

- Upon the request of CM for ROS to perform deleting instance of RA, ROS deletes the instance of designated RA.
- Upon completion of deleting instance, CM acknowledges deleting instance of RA to MPM using *DeInstantiateRACnf* signal.

6.2 Procedures for list checking of RA(s)

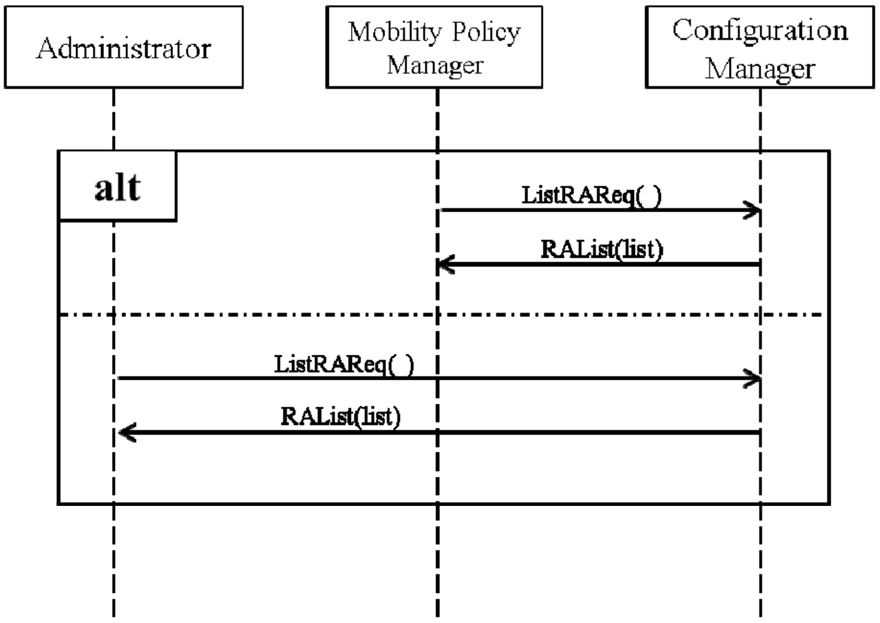

Figure 6.3: Signalling diagram of list checking the RA(s)

Figure 6.3 illustrates a signalling diagram showing the procedure of list checking of RA(s).

The procedure of checking the list of installed/instantiated/active RA(s) shown in Figure 6.3 can be summarized as follows:

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

6.3 Procedures for activation/deactivation of RA

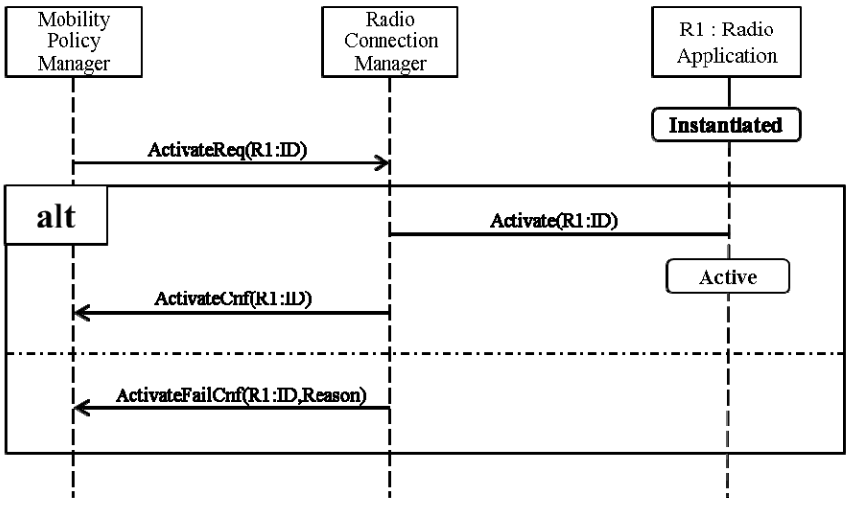

Figure 6.4: Signalling diagram of activating RA

Figure 6.4 illustrates a signalling diagram showing the procedure of activating RA.

The procedure of activating RA shown in Figure 6.4 can be summarized as follows:

- MPM transfers *ActivateReq* signal including ID of RA to request RA activation to RCM.
- Upon the request of RCM for ROS to perform RA activation, ROS activates the designated RA.
- After ROS completes the activation of RA, RCM acknowledges the completion of RA activation by sending *ActivateCnf* signal to MPM.
- If RA activation is failed, RCM reports the failure of RA activation to MPM by transferring the failed RA ID and failure reason in *ActivateFailCnf* signal.

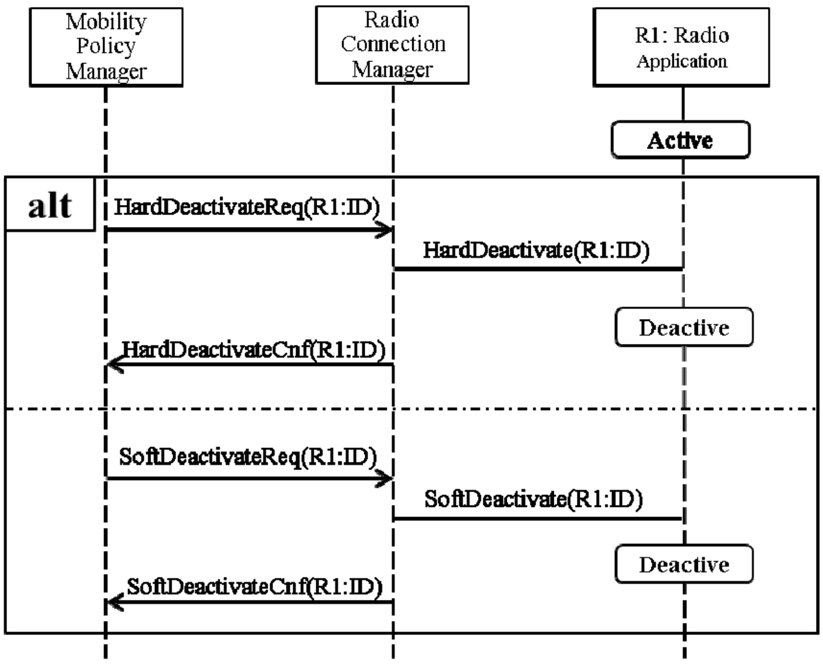

Figure 6.5: Signalling diagram of deactivating RA

Figure 6.5 illustrates a signalling diagram of deactivating procedure.

The procedure of deactivating shown in Figure 6.5 can be summarized as follows:

- In the case of hard deactivation, MPM transfers *HardDeactivateReq* signal including ID of RA to request hard deactivation of the designated RA to RCM.
- Upon the request of RCM for ROS to perform hard deactivation of RA, ROS deactivates the designated RA.
- After ROS completes the hard deactivation of RA, RCM acknowledges the completion of hard deactivation of RA by sending *HardDeactivateCnf* signal to MPM.
- In the case of soft deactivation, MPM transfers *SoftDeactivateReq* signal including ID of RA to request soft deactivation of the designated RA to RCM.
- Upon the request of RCM for ROS to perform soft deactivation of RA, ROS deactivates the designated RA.
- After ROS completes the soft deactivation of RA, RCM acknowledges the completion of soft deactivation of RA by sending *SoftDeactivateCnf* signal to MPM.

6.4 Procedures for transferring context information

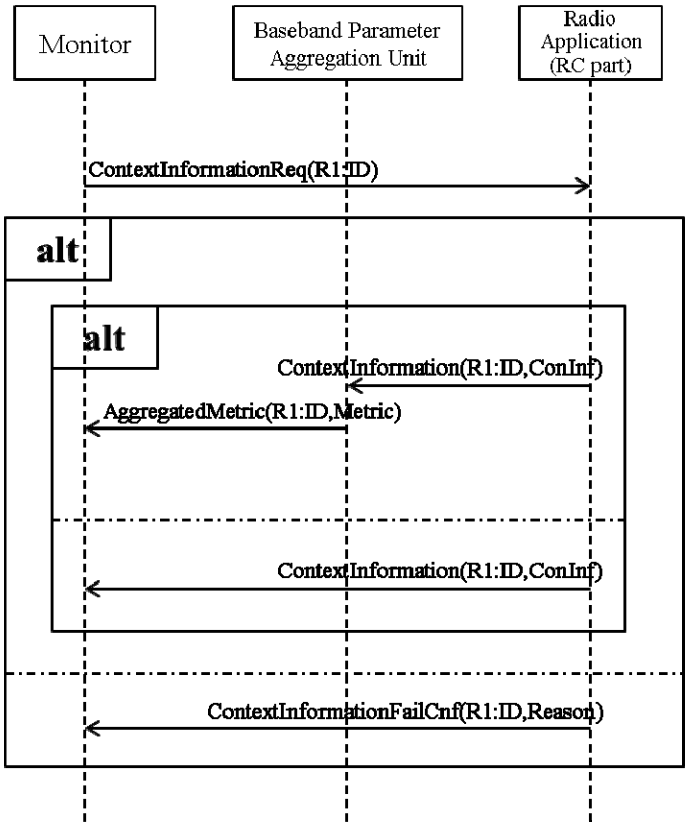
Monitor
Baseband Parameter Aggregation Unit
Radio Application (RC part)
ContextInformationReq(R1:ID)
alt
alt
ContextInformation(R1:ID,ConInf)
AggregatedMetric(R1:ID,Metric)
ContextInformation(R1:ID,ConInf)
ContextInformationFailCnf(R1:ID,Reason)

Figure 6.6: Signalling diagram of transferring context information

Figure 6.6 illustrates a signalling diagram of transferring context information.

The procedure of transferring the context information shown in Figure 6.6 can be summarized as follows:

- Monitor transfers *ContextInformationReq* signal including RA ID to RC in RA.
- RC in RA transfers *ContextInformation* signal including RA ID and context information generated in corresponding function block(s) in RA to Monitor.
- In the case of using Baseband Parameter Aggregation (BPA) unit, RC in RA transfers *ContextInformation* signal including RA ID and context information to BPA unit. Then the BPA unit aggregates and compresses the context information to minimize the bandwidth occupied by the context information to be transferred. Upon completion of procedure of context information aggregation and compression, BPA unit transfers *AggregatedMetric* signal including RA ID and aggregated metric(s) to Monitor.
- In the case of generating context information failure, RC in RA transfers *ContextInformationFailCnf* signal including to RA ID and failure reason to Monitor.

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

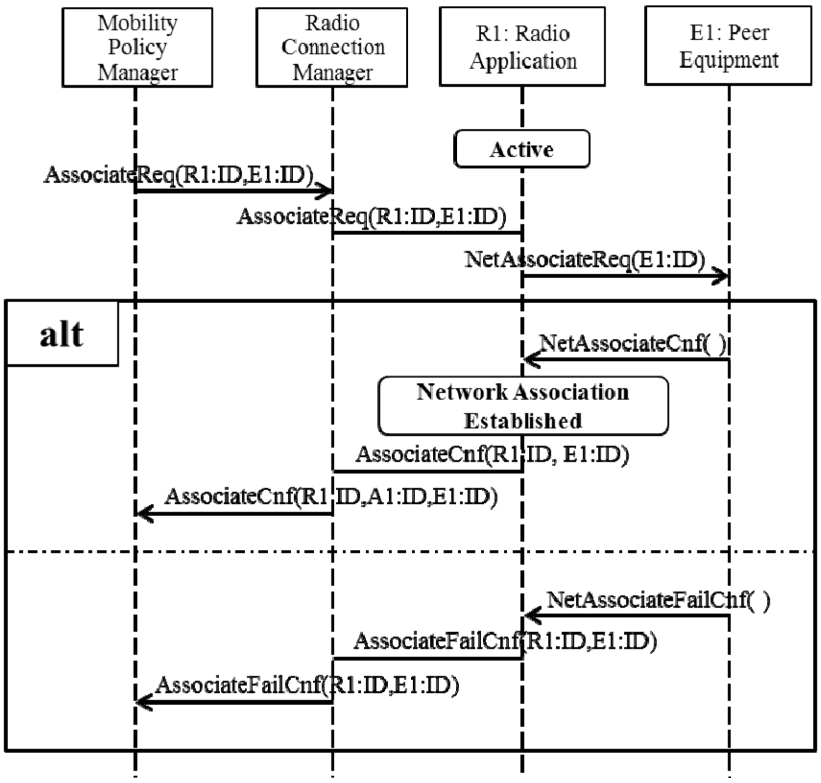
Mobility Policy Manager
Radio Connection Manager
R1: Radio Application
E1: Peer Equipment
Active
AssociateReq(R1:ID,E1:ID)
AssociateReq(R1:ID,E1:ID)
NetAssociateReq(E1:ID)
alt
NetAssociateCnf( )
Network Association Established
AssociateCnf(R1:ID, E1:ID)
AssociateCnf(R1:ID,A1:ID,E1:ID)
NetAssociateFailCnf( )
AssociateFailCnf(R1:ID,E1:ID)
AssociateFailCnf(R1:ID,E1:ID)

Figure 6.7: Signalling diagram of creating network association

Figure 6.7 illustrates a signalling diagram of creating network association procedure.

The procedure of creating network association shown in Figure 6.7 can be summarized as follows:

- MPM transfers *AssociateReq* signal including RA ID and peer equipment 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. [i.2].
- Upon the request of RCM for ROS to perform creating network association, ROS transfers *AssociateReq* signal from RCM to RA. Then, RA transfers ID of corresponding peer equipment using *NetAssociateReq* signal.
- Upon completion of creating network association, peer equipment transfers *NetAssociateCnf* signal to RA. Then ROS transfers *AssociateCnf* signal to RCM, which transfers *AssociateCnf* signal to MPM.
- In the case of creating network association failure, peer equipment transfers *NetAssociateFailCnf* signal to RA. Then ROS transfers *AssociateFailCnf* signal to RCM, which transfers *AssociateFailCnf* signal to MPM.

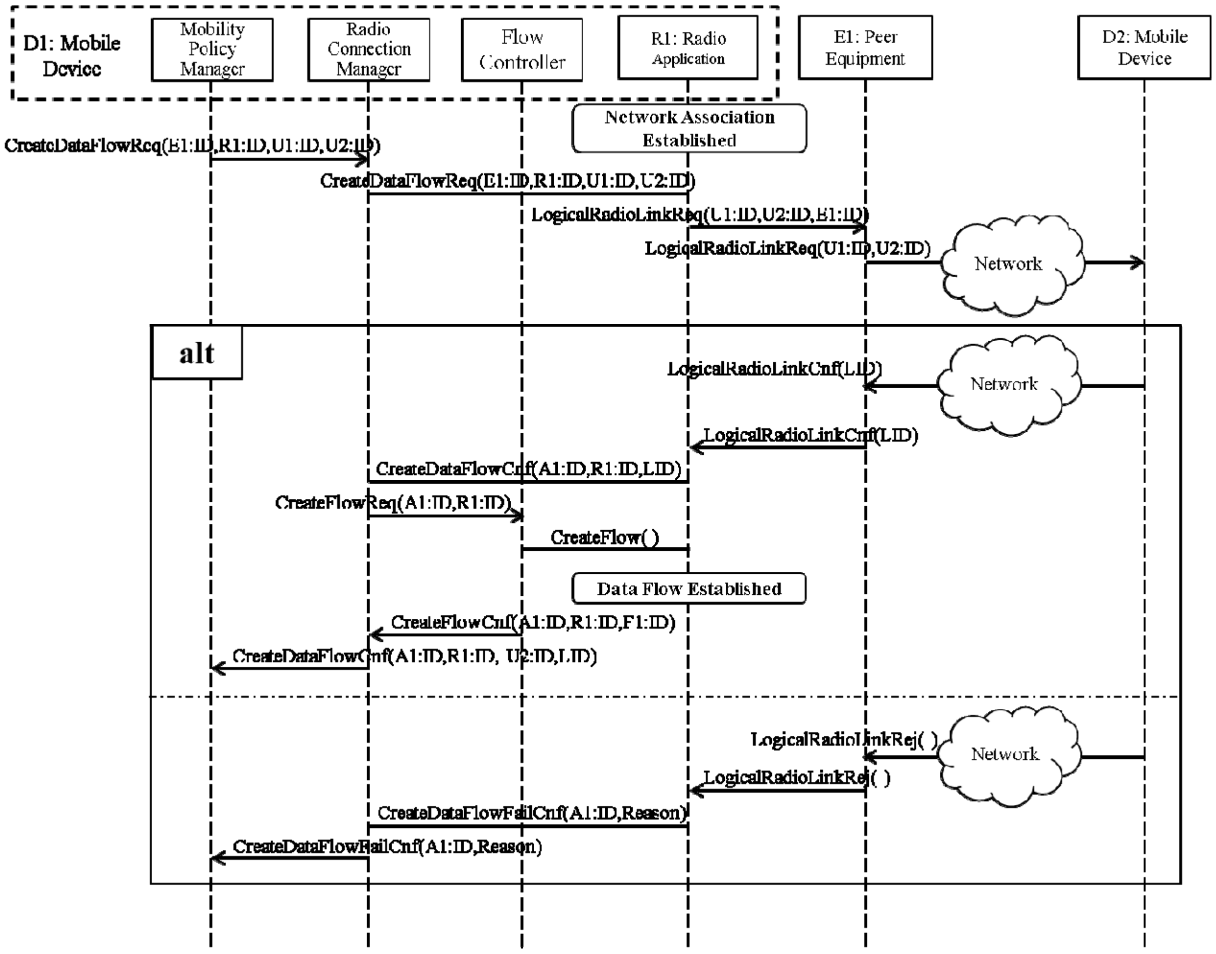
D1: Mobile Device
Mobility Policy Manager
Radio Connection Manager
Flow Controller
R1: Radio Application
E1: Peer Equipment
D2: Mobile Device
Network Association Established
CreateDataFlowReq(E1:ID,R1:ID,U1:ID,U2:ID)
CreateDataFlowReq(E1:ID,R1:ID,U1:ID,U2:ID)
LogicalRadioLinkReq(U1:ID,U2:ID,E1:ID)
LogicalRadioLinkReq(U1:ID,U2:ID)
Network
alt
LogicalRadioLinkCnf(LID)
Network
LogicalRadioLinkCnf(LID)
CreateDataFlowCnf(A1:ID,R1:ID,LID)
CreateFlowReq(A1:ID,R1:ID)
CreateFlow( )
Data Flow Established
CreateFlowCnf(A1:ID,R1:ID,F1:ID)
CreateDataFlowCnf(A1:ID,R1:ID, U2:ID,LID)
LogicalRadioLinkRej( )
Network
LogicalRadioLinkRej( )
CreateDataFlowFailCnf(A1:ID,Reason)
CreateDataFlowFailCnf(A1:ID,Reason)

Figure 6.8: Signalling diagram of creating logical radio link association

Figure 6.8 illustrates a signalling diagram of creating logical radio link association procedure.

The procedure of creating logical radio link association shown in Figure 6.8 can be summarized as follows:

- MPM transfers *CreateDataFlowReq* signal to RCM including peer equipment ID, active RA ID, user ID and another MD user ID in order to associate logical radio link with another MD.
- RCM requests ROS to create data flow using *CreateDataFlowReq* signal including peer equipment ID, active RA ID, user ID and another MD user ID. Then, RA transfers ID of corresponding user ID, another MD user ID, and peer equipment ID using *LogicalRadioLinkReq* signal to peer equipment. Upon receiving *LogicalRadioLinkReq* signal including user ID and another MD user ID from peer equipment, network transfers *LogicalRadioLinkCnf* signal including logical link ID to peer equipment.
- Upon transferring *LogicalRadioLinkCnf* signal including logical link ID from peer equipment to RA, ROS transfers *CreateDataFlowCnf* signal including network association ID, RA ID, and logical link ID to RCM.
- In order to set up data flow, RCM transfers *CreateFlowReq* signal including network association ID and RA ID to FC. After creating data flow, FC transfers *CreateFlowCnf* signal including network association ID, RA ID, and created data flow ID to RCM.
- RCM transfers *CreateDataFlowCnf* signal including network association ID, RA ID, and data flow ID to MPM.
- When RA receives *LogicalRadioLinkRej* signal from the peer equipment, ROS transfers *CreateDataFlowFailCnf* signal including network association ID and failure reason to RCM, which transfers the *CreateDataFlowFailCnf* signal to MPM to acknowledge the failure of creating data flow.

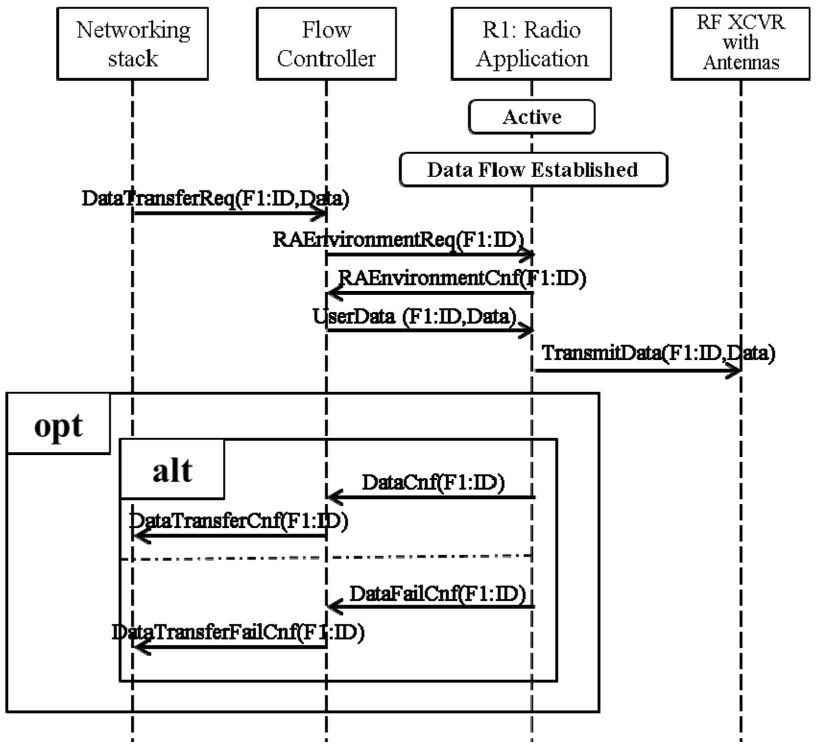

Figure 6.9: Signalling diagram of sending data

Figure 6.9 illustrates a signalling diagram of sending data procedure.

The procedure of sending data shown in Figure 6.9 can be summarized as follows:

- Networking stack transfers *DataTransferReq* signal together with data flow ID and user data to FC in order to transfer user data.
- FC transfers *RAEnvironmentReq* signal to RA in order to request information about user data to be transferred to RA such as throughput, data bandwidth, etc.
- RA transfers environmental information using *RAEnvironmentCnf* signal to FC.
- Upon receiving *RAEnvironmentCnf* signal including data flow ID, FC transfers *UserData* signal together with data flow ID and user data to RA.
- RA transfers user data including data flow ID using *TransmitData* signal to RF XCVR with antenna(s).
- After completion of sending data, RA acknowledges the completion of sending user data by transferring *DataCnf* signal to FC.
- Upon receiving *DataCnf* signal, FC transfers *DataTransferCnf* signal together with data flow ID to Networking stack.
- In the case of sending data failure, RA reports the failure of sending data to FC by transferring *DataFailCnf* signal including data flow ID.
- Upon receiving *DataFailCnf* signal, FC transfers *DataTransferFailCnf* signal together with data flow ID to Networking stack.

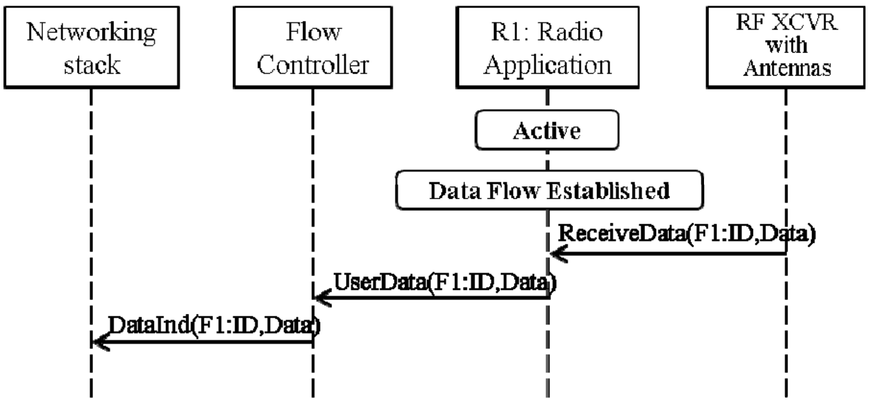

Figure 6.10: Signalling diagram of receiving data

Figure 6.10 illustrates a signalling diagram of receiving data procedure.

The procedure of receiving data shown in Figure 6.10 can be summarized as follows:

- RF XCVR with antenna(s) transfers received user data including data flow ID using *ReceiveData* signal to RA.
- RA transfers *UserData* signal including data flow ID and user data to FC after decoding the data received from RF XCVR with antenna(s).
- FC transfers *DataInd* signal including data flow ID and user data received from RA to Networking stack.

7 Conclusion

In the present document, we suggested the radio reconfiguration related architecture for Reconfigurable MD. We also introduced the procedures of MD operations performed with the reference points.

- Overview of radio reconfiguration related architecture reference model for MD:
  - Detailed substances of Reconfigurable MD architecture for multiradio applications, software architecture for RP and operational structure of URA are given in clause 4.
- Reference Points:
  - Detailed substances of interfaces for installation/uninstallation and creating/deleting instance of RA, list checking of RA(s), activation/deactivation of RA, transferring context information, creating data flow, and sending/receiving user data are given in clause 5.
- Procedures:
  - Detailed substances of procedures for installation/uninstallation and creating/deleting instance of RA, list checking of RA(s), activation/deactivation of RA, transferring context information, creating data flow, and sending/receiving user data are given in clause 6.

Based on Reconfigurable MD architecture, reference points and procedures provided in the present document, ETSI Technical Specifications will be generated as follows:

- Normative Protocol and Interfaces Specification (TS).

History

Document history		
V1.1.1	January 2013	Publication