WInnForum Facilities Principles

Eric Nicollet (Thales)

Presentation at WInnComm 2021
1 Dec 2021 – Online event

Agenda

Introduction

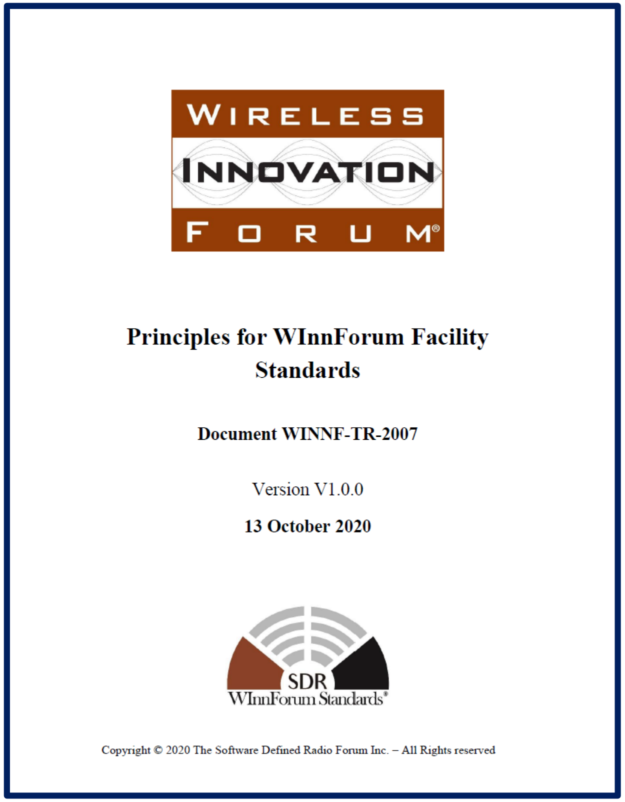
Principles for WInnForum Facility Standards

WInnForum Facility PSMs Mapping Rules

Introduction

- WInnForum developed a formally structured framework for Facility specifications
- Two technical report capture this structuring
  - TR-2007 *Principles for WInnForum Facility Standards*
  - TR-2008 *WInnForum Facility PSMs Mapping Rules*
- Consistent with 2 Facility specifications under finalization
  - TS-0008 Transceiver Facility V2.1
  - TS-3004 Time Service Facility V1.1

Technical Report "*Principles for WInnForum Facility Standards*"


Principles for WInnForum Facility Standards

Document WINNF-TR-2007

Version V1.0.0

13 October 2020

SDR
WInnForum Standards®

Copyright © 2020 The Software Defined Radio Forum Inc. – All Rights reserved

Sets the reference concepts for specificaiton of WInnForum Facilities

- WINNF-TR-2007 V1.0
- October 2020
- 13 pages

Introduction (§ 1)

D01 A WInnForum *facility* **is defined as** a WInnForum specification that applies the "*Principles for WInnForum Facility Standards*".

Tenets of a Facility Specification

- Addresses *functional support capabilities* (e.g., transceiver, timing service, audio),
- Service-oriented approach,
- OMG Model Driven Architecture (MDA) paradigm,
- Specification of one PIM and several PSMs,
- Specification of services, associated API and attributes,
- Flexibility and scalability thanks to formalized optionality model.

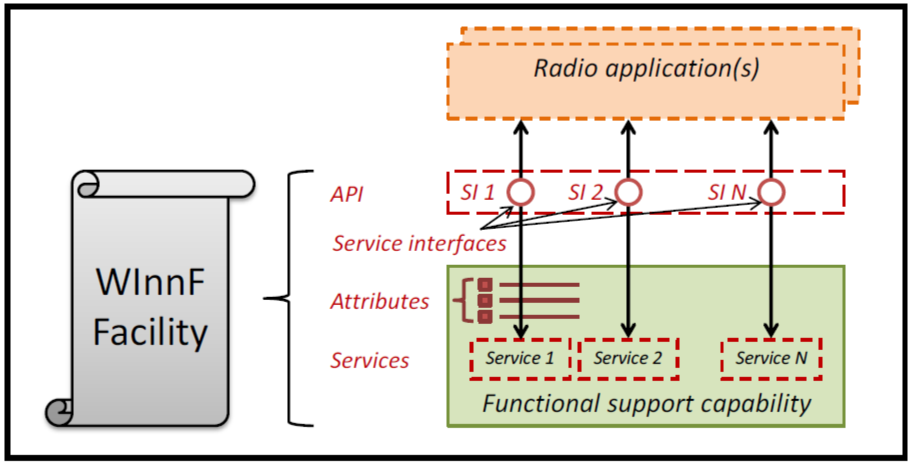


Figure 1 WInnForum facility overview

Driving the future of radio communications and systems worldwide

General principles (§ 2)

Software Defined Radio

D02 A *radio capability* **is defined as** a capability available on a radio product based on over-the-air radio operation (transmit-receive, transmit-only or receive-only).

D03 A *software defined radio* **is defined as** a radio that implements *radio capabilities* through execution of software applications.

D04 A *radio application* **is defined as** a software application instance that implements a *radio capability* within a *software defined radio*.

D05 A *radio platform* **is defined as** the hardware and software environment provided by a *software defined radio* for execution of *radio applications*.

Benefits of SDR Standards

D06 The *portability* concept **is defined as**, for a *radio application*, the level of reduction of effort in having an existing *radio application* running on new *radio platform*.

D07 The *hospitality* concept **is defined as**, for a *radio platform*, the level of reduction of effort in having a *radio application* running on that *radio platform*.

- Improving *portability* and *hospitality*

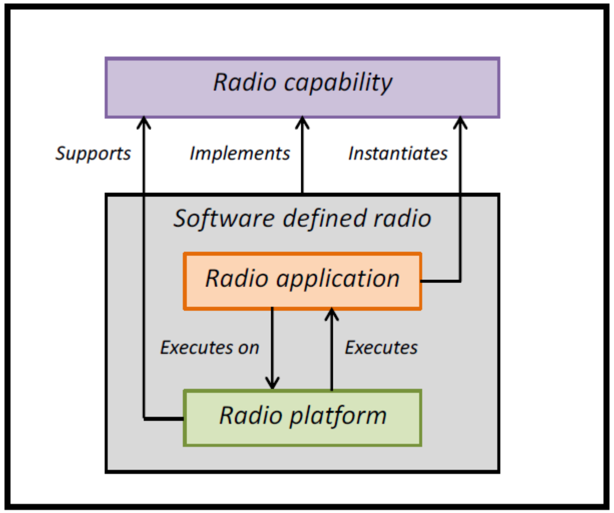


Figure 2 Base concepts

SDR technical principles (§ 3)

D08 An *application component* **is defined as** a software component of a *radio application*.

D09 A *processing node* **is defined as** a processor of the *radio platform* capable to execute *application components*.

§ 3.1 Architecture concepts

- Component-based *radio applications*
- Need to address a large variety of *processing nodes*: GPP, DSP, FPGA…

Software support (§ 3.1.2)

D10 The *software support* **is defined as** the capabilities of a *radio platform* that enable execution of *application components* throughout the available *processing nodes*.

D11 A *software environment* **is defined as** the capabilities of a given *processing node* that enable execution of *application components*.

Example constituents

- Scheduling (e.g. POSIX)
- Connectivity (e.g. CORBA)
- Components handling (e.g. SCA CF)

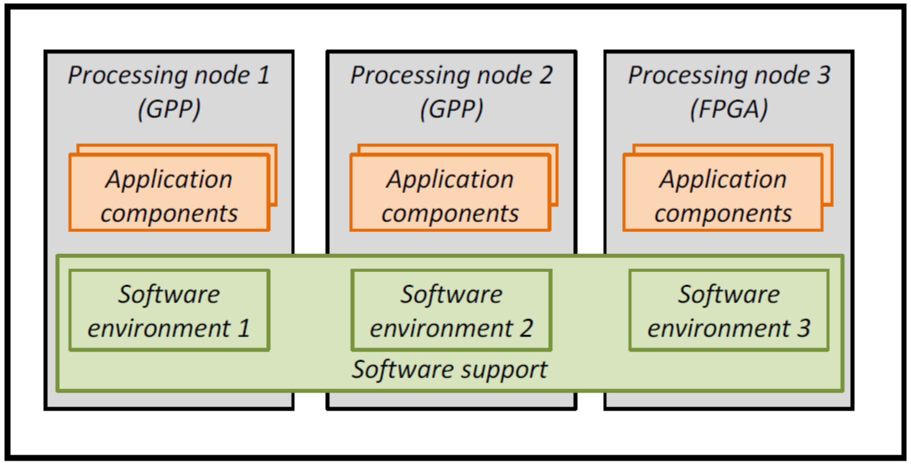


Figure 3 Software support

Functional support (§ 3.1.3)

D12 The *functional support* **is defined as** the capabilities of a *radio platform* that provide functionalities specific to the radio domain in support of *application components*.

D13 A *functional support capability* **is defined as** one elementary capability of the *functional support*.

D14 A *façade* **is defined as** the software segment of a *functional support capability* implementation that executes on a given *processing node*.

D15 An *access paradigm* **is defined as** the software mechanisms enabling an *application component* to access to a *façade* within the concerned *processing node*.

Examples

- Transceiver
- Time service
- GNSS
- Audio port
- Serial port
- Pseudo-random noise generator

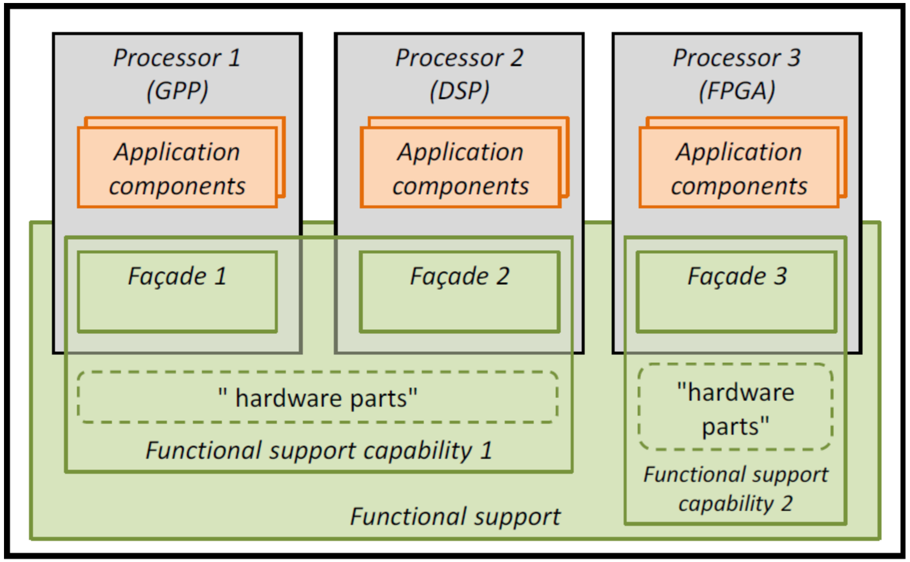


Figure 4 Functional support

Service-oriented functional support (§ 3.1.2)

D16 A *service* **is defined as** one elementary capability provided by a *functional support capability* to *radio applications*.

D17 A *service name* **is defined as** the name of a *service*.

D18 A *service implementation* **is defined as** an implementation of a particular *service* by a particular *façade*.

D19 A *service interface* **is defined as** the software interface presented by a *service* to the *radio application(s)* employing it.

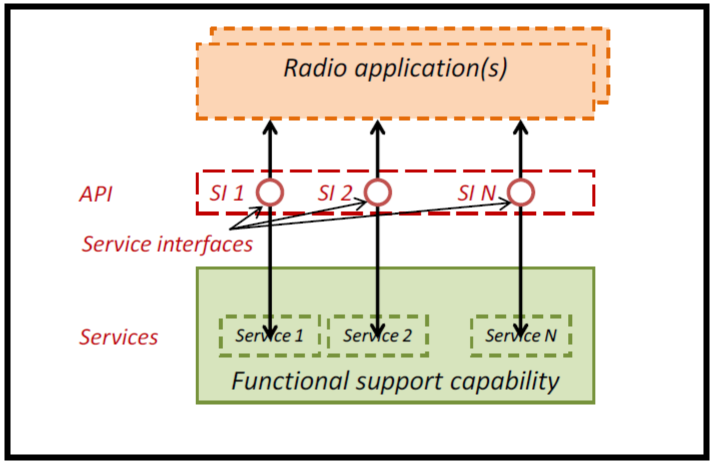


Figure 5 Services

Provide and use services (§ 3.2.2)

D20 A *provide service* **is defined as** a *service* whose *service interface* is used by *radio applications* and provided by a *functional support capability*.

D21 A *use service* **is defined as** a *service* whose *service interface* is used by a *functional support capability* and provided by *radio applications*.

D22 A *services group* **is defined as** a consistent set of *use services* and *provide services* of a *functional support capability* that answers to a common use case.

D23 A *services group name* **is defined as** the name of a *services group*.

D24 A *primitive* **is defined as** one of the primitives composing a *service interface*.

D25 A *primitive implementation* **is defined as** an implementation of a particular *primitive* within a *service implementation*.

The following software *engineering* concepts are attached to *primitives*:

- **D26** *signature*,
- **D27** *parameter*,
- **D28** *direction* (“in”, “out”, “inout” indicator),
- **D29** *semantics* of:
  - *parameters* (meaning and behaviors attached to *parameters*),
  - *primitives*,
- **D30** *type*,
- **D31** *exception*.

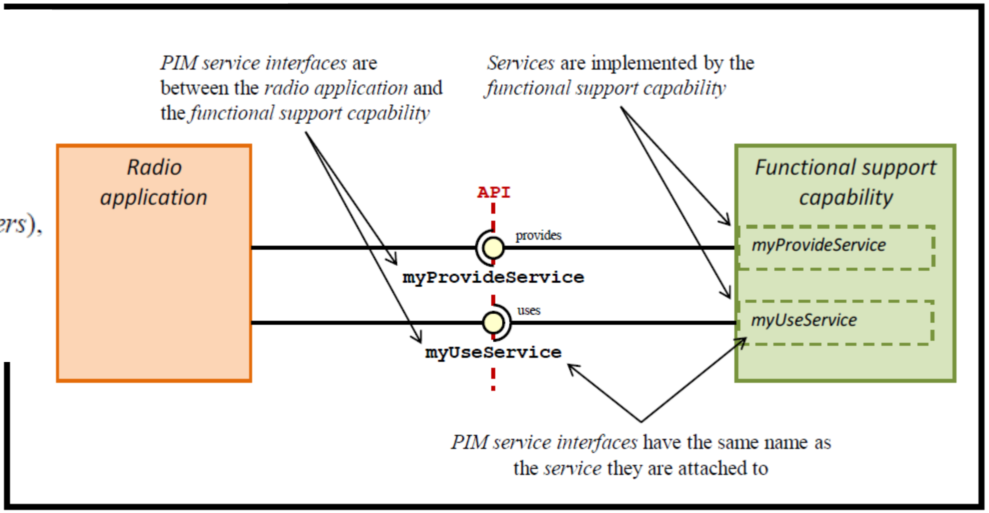


Figure 6 Services orientation

Real-time concepts (§ 3.2.5)

D32 The *call time* of a *primitive implementation* **is defined as** the instant when it is called.

t_{call} **denotes** the *call time* of a *primitive implementation*.

D33 The *return time* of a *primitive implementation* **is defined as** the instant when it returns.

t_{return} **denotes** the *return time* of a *primitive implementation*.

D34 The *worst-case execution time* (*WCET*) of a *primitive implementation* of a *provide service* **is defined as** the maximum time taken by the implementation between its *call time* and *return time*.

D35 The *worst-case external execution time* (*WCEET*) of a *primitive implementation* of a *use service* **is defined as** the maximum time supported by the implementation between t_{call} and t_{return}.

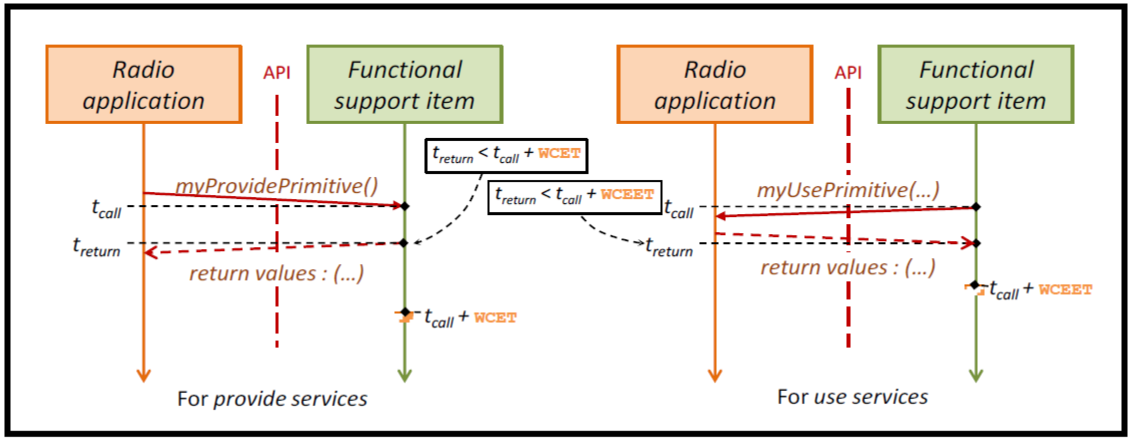


Figure 7 Services primitives call and return time

Facility attributes (§ 3.3)

D36 A *facility attribute* **is defined as** an object-oriented attribute of a *functional support capability* that conditions its correct joint execution with a *radio application*.

Examples

- Behavioral option
- Transfer function
- Set of supported services
- Real-time performance values

Counter-examples

- SWaP of implementations
- Any other feature with no impact on the *radio application*

Categories

D37 A *capability* **is defined as** a *facility attribute* constant over the lifetime of a *functional support capability* implementation.

D38 A *property* **is defined as** a *facility attribute* constant over the configured state of a *functional support capability* implementation.

D39 A *variable* **is defined as** a *facility attribute* of a *functional support capability* implementation that is not meant to be constant.

WIRELESS INNOVATION FORUM

Driving the future of radio communications and systems worldwide

SDR forum version 2.0

Specification principles (§ 4)

A *facility* is composed of a PIM (Platform-Independent Model) specification completed by derived PSM (Platform-Specific Model) specifications.

PIM specification

D40 A *PIM specification* **is defined as** a specification that answers to the definition of a PIM provided by [Ref2]: "A PIM exhibits a sufficient degree of independence so as to enable its mapping to one or more platforms. This is commonly achieved by defining a set of services in a way that abstracts out technical details. Other models then specify a realization of these services in a platform specific manner.".

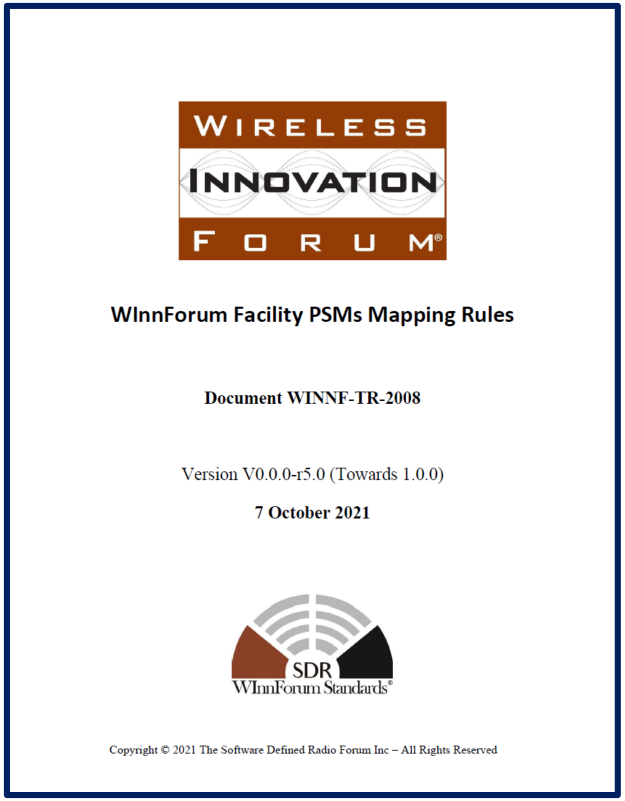
A *PIM specification* uses the WInnForum "*IDL Profiles for Platform-Independent Modeling of SDR Applications*" [Ref3] to specify the *service interfaces* of the *functional support capability*.

This is consistent with usage of SCA 4.1 Appendix E-1 "*Application Interface Definition Language Platform Independent Model Profiles*" (see [Ref4]).

PSM specification

D41 A *PSM specification* **is defined as** a specification that answer to the definition of a PSM provided by [Ref2]: "A PSM combines the specifications in the PIM with the details required to stipulate how a system uses a particular type of platform. If the PSM does not include all of the details necessary to produce an implementation of that platform it is considered abstract (meaning that it relies on other explicit or implicit models which do contain the necessary details).".

Technical Report "*WInnForum Facility PSMs Mapping Rules*"



WIRELESS INNOVATION FORUM

WInnForum Facility PSMs Mapping Rules

Document WINNF-TR-2008

Version V0.0.0-r5.0 (Towards 1.0.0)

7 October 2021

SDR WInnForum Standards

Copyright © 2021 The Software Defined Radio Forum Inc – All Rights Reserved

Specifies mapping rules for 3 *programming paradigms*

- Native C++
- SCA
- FPGA

- WINNF-TR-2008 V1.0
- Early 2022 (in approval)
- 42 pages

Reference architectural pattern (§ 1.2)

Reference architectural pattern

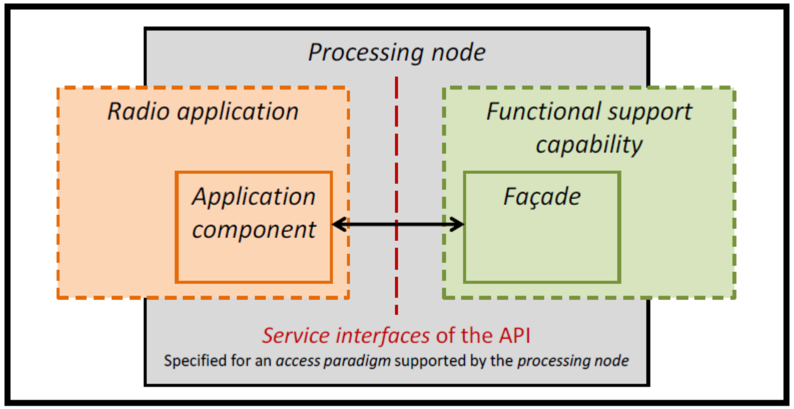


Figure 1 Reference architectural pattern

A *radio application* is possibly composed of a number of *application components* distributed across a composition of *processing nodes*.

The *radio platform* implements a number *functional support capabilities*, accessible on a number of *processing nodes* through software interfaces presented by *façades*.

The *façades* are the software parts of a *functional support capability* implementation that present a number of *service interfaces* for employment by *application components*.

The set of *service interfaces* supported by a *façade* belong to the API specified by the *PIM specification* of the considered *functional support capability*, and are derived by the *PSM specification* according to the applied *access paradigm*.

Nothing prevents a given *processing node* to support more than one *access paradigm*.

Native C++ (§ 2)

The *native C++ access paradigm* **is defined as** an *access paradigm* based on direct native C++ connection between *application components* and *façades*.

It is based on two C++ versions: C++ 11 (see [Ref2]) and C++ 2003 (see [Ref3]).

A *native C++ PSM specification* **is defined as** a standard specifying, according to the *native C++ access paradigm*, interfaces between instances of *radio applications* and instances of the addressed *functional support capability*.

A *native C++ application component* can:

- Be a component of the *radio application* running in the same *native C++ node*,
- A proxy of a component of the *radio application* running in a remote *processing node*.

In the proxy case, the remote component complies with a *PSM specification* that may be:

- The *native C++ PSM specification*, if the remote *processing node* is another *native C++ node*,
- Another *PSM specification*, if the remote *processing node* is not a *native C++ node*.

The proxy uses a connectivity mechanism between the *native C++ node* and the remote *processing node* that can typically be standard compliant (e.g. MHAL Communication Service, MOCB, CORBA), or be a proprietary solution.

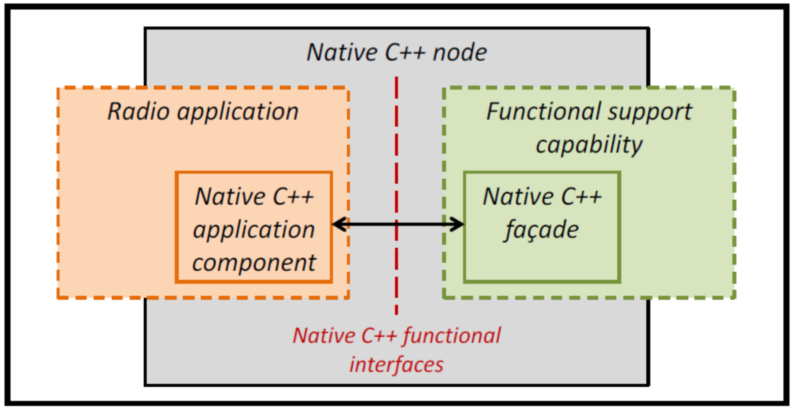


Figure 2 Positioning of *native C++ functional interfaces*

A *native C++ façade* **is conformant with** the *native C++ PSM specification* of a *functional support capability* if it provides an implementation of the `Facade` class and its related *service interfaces*.

A *native C++ application component* **is conformant with** the *native C++ PSM specification* if it can use *native C++ façades* conformant with the *native C++ PSM specification*, without using any non-standard *service interface* for the *functional support capability*.

The Facade class (§ 2.6)

The **Facade** class **is specified as** a class providing *native C++ application components* with access to *native C++ façades*.

For *functional support capabilities* featuring the **CONFIGURED** state, the **Facade** class owns *activeServicesInitialized()* and *activeServicesReleased()* methods.

The **Facade** class also owns at least one of the following interfaces for *services* access: **ExplicitServicesAccess** (see section 2.6.4) or **GenericServiceAccess** (see section 2.6.5).

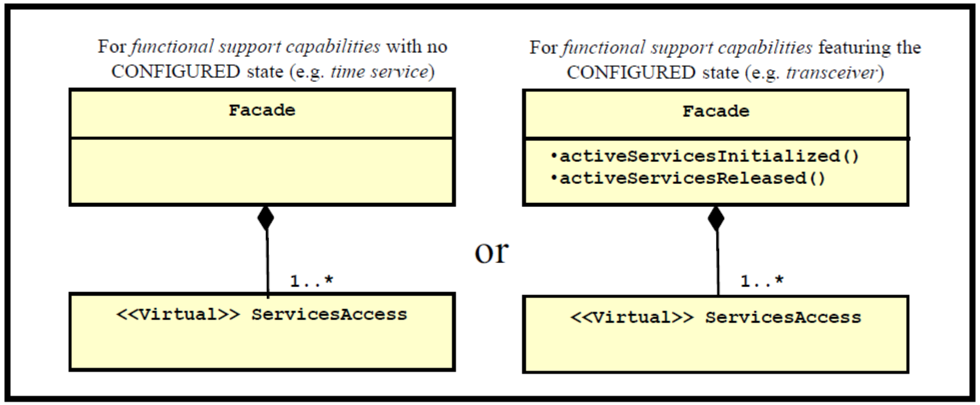


Figure 3 Class diagram of a *native C++ façade*

Explicit or generic services access

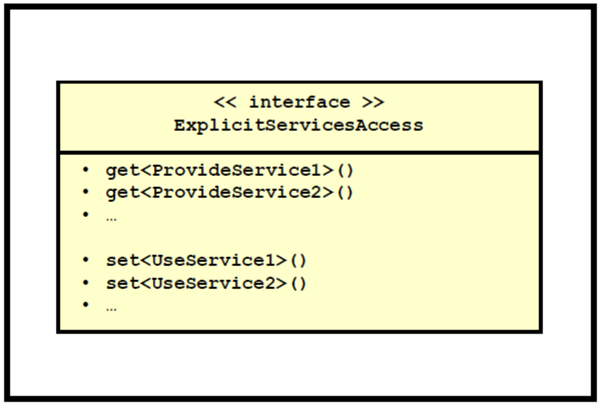


Figure 4 Class diagram of *explicit services access*

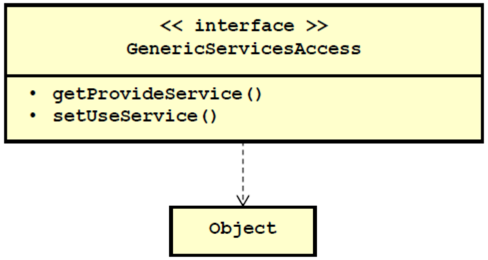


Figure 5 Class diagram of *generic services access*

SCA (§ 3)

The *SCA access paradigm* **is defined as** an *access paradigm* based on SCA connections between *application components* and *façades*.

It is based on two SCA versions: SCA 2.2.2 (see [Ref7]) and SCA 4.1 (see [Ref8]).

An *SCA PSM specification* **is defined as** a standard specifying, according to the *SCA access paradigm*, interfaces between instances of *radio applications* and instances of the addressed *functional support capability*.

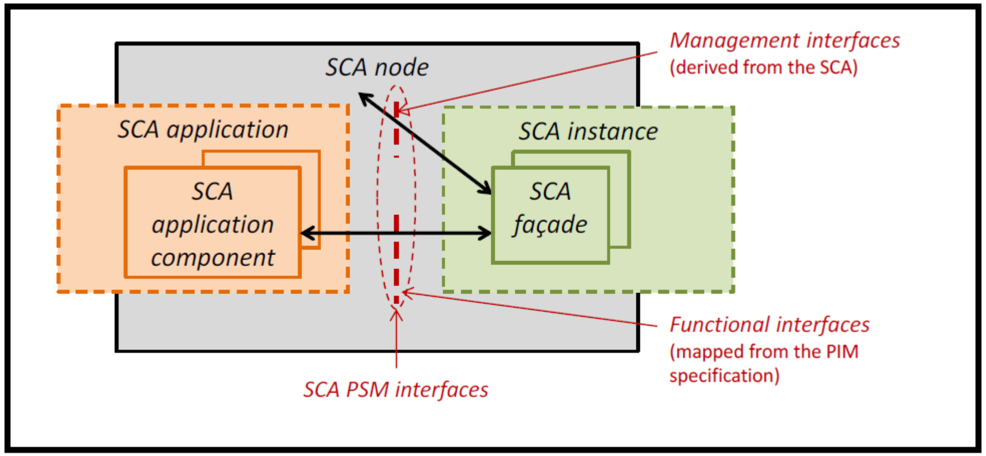


Figure 6 Architecture concepts for SCA PSMs

An *SCA façade* **is conformant with** the *SCA PSM specification* of a *functional support capability* if it provides an SCA implementation of *service interfaces*.

An *SCA application component* **is conformant with** the *SCA PSM specification* of a *functional support capability* if it can use *SCA façades* conformant with the *SCA PSM specification*, without using any non-standard *service interface* for the *functional support capability*.

SCA 2.2.2 Management Interfaces

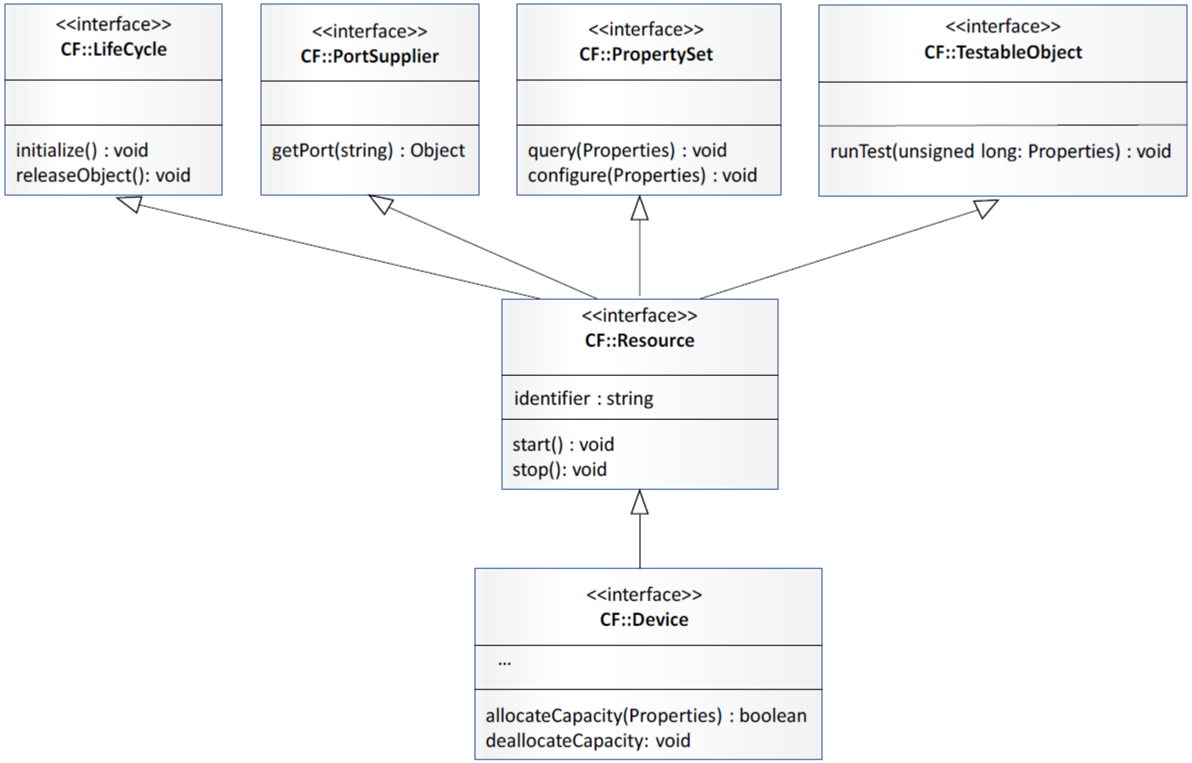


Figure 7 ***SCA 2.2.2 PSM management interfaces***

SCA 4.1 Management Interfaces

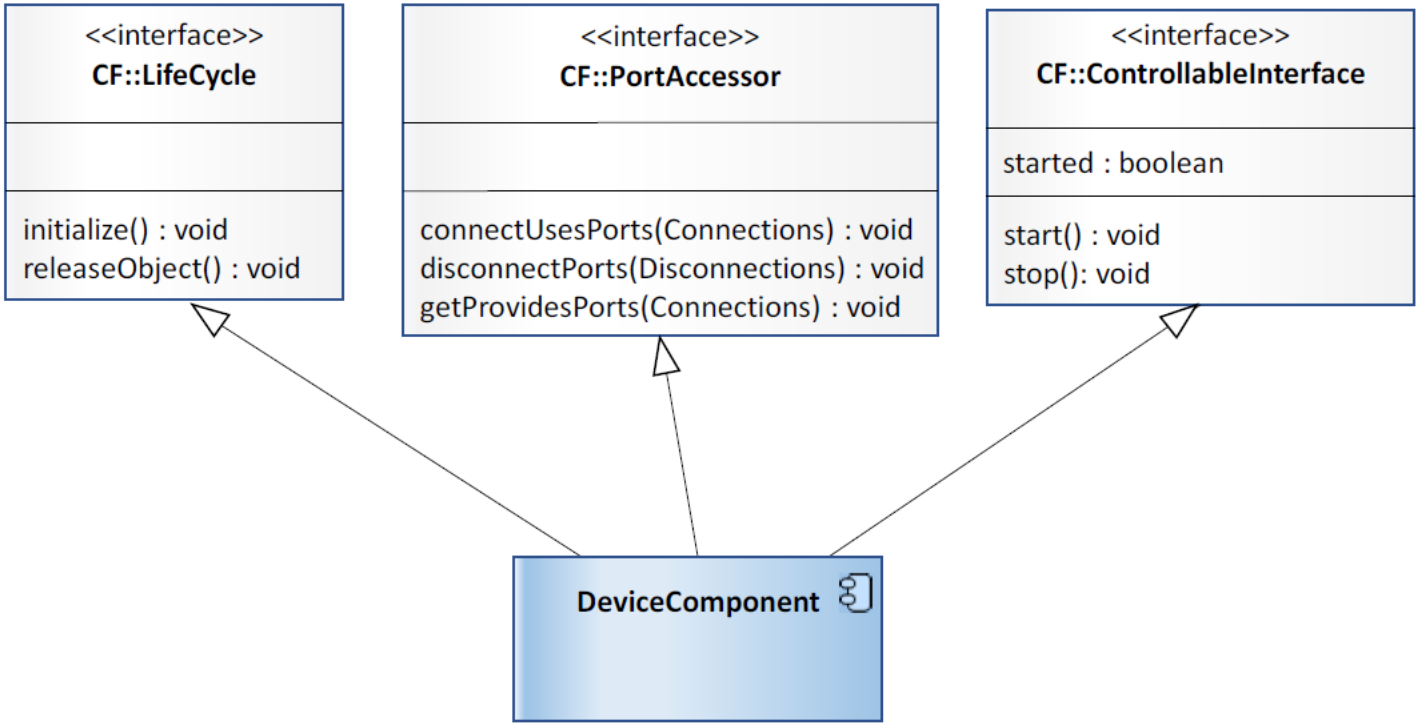


Figure 8 ***SCA 4.1 PSM management interfaces***

FPGA (§ 4)

FPGA functional interfaces **are defined as** the FPGA interfaces derived from the *service interfaces* of a *PIM specification*.

An *FPGA PSM specification* **is defined as** a specification that standardizes *FPGA functional interfaces* between instances of *radio applications* and *functional support capabilities*.

An *FPGA node* **is defined as** an FPGA of a *radio platform* providing *radio applications* with *FPGA functional interfaces* related to one or several *functional support capabilities*.

An *FPGA façade* **is defined as** a *façade* of a *functional support capability* instance that executes within an *FPGA node*.

An *FPGA applicative module* **is defined as** a module of a *radio application* implemented in an *FPGA node* that employs at least one *FPGA façade*.

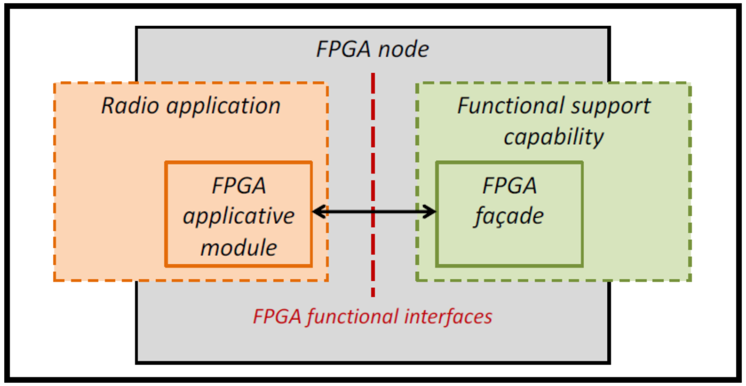


Figure 9 Positioning of *FPGA functional interfaces*

The *FPGA applicative module* can:

- Be a component of the *radio application* running in the same *FPGA node*,
- A proxy of a component of the *radio application* running in a remote *processing node*.

In the proxy case, the remote component conforms with a *PSM specification* that may be:

- The *FPGA PSM specification*, if the remote *processing node* is another *FPGA node*,
- Another *PSM specification*, if the remote *processing node* is not an *FPGA node*.

The proxy uses a connectivity mechanism between the *FPGA node* and the remote *processing node* that can typically be a standard (e.g. MHAL Communication Service, MOCB), an FPGA extension of CORBA, or a proprietary solution.

RTL signals

RTL signal name <FSC_TAG>_<instNum>_<PRIM_NAME>_+	Origin	Format	Specification
CLK	*FPGA façade*	1-bit signal	Clock attached to the *FPGA primitive*.
RST	*FPGA façade*	1-bit signal	Hardware reset propagation to the *FPGA primitive*.

Table 16 ***Structural RTL signals***

RTL signal name <FSC_TAG>_<instNum>_<PRIM_NAME>_+	Origin	Format	Usage case	Specification
EN	*Caller*	1-bit signal	No in parameter and no explicit return.	The *FPGA primitive* is called.
RDY	*Callee*	1-bit signal	Blocking behavior.	The *callee* is ready to receive a new call on the *FPGA primitive*.

Table 17 ***Semantics RTL signals***

RTL signal name <FSC_TAG>_<instNum>_<PRIM_NAME>_+	Origin	Format		
EN_IN	*Caller*	1-bit signal	in ... explicit return.	... is called. Validates in param(s).
DATA_IN.<param_n>	*Caller*	param_n format	in param(s).	Value of n^{th} in param.
EN_OUT	*Callee*	1-bit signal	Explicit return.	The *FPGA primitive* returns. Validates out param(s).
DATA_OUT.<param_n>	*Callee*	param_n format	out param(s).	Value of n^{th} out param.

Table 18 ***Parameters RTL signals***

What's next?

Please refer to focused presentations on

Transceiver Facility

Time Service Facility

Will be ruled out early 2022

Selection of the *functional support item* for next Facility specification effort under progress

End of the presentation
Thank you for your attention

Any questions?

Contact:

- eric.nicollet@thalesgroup.com