WInnForum Time Service Facility and SOSA™: Alignment with Objectives

WInnForum Webinar Series #32

Webinar Administrivia

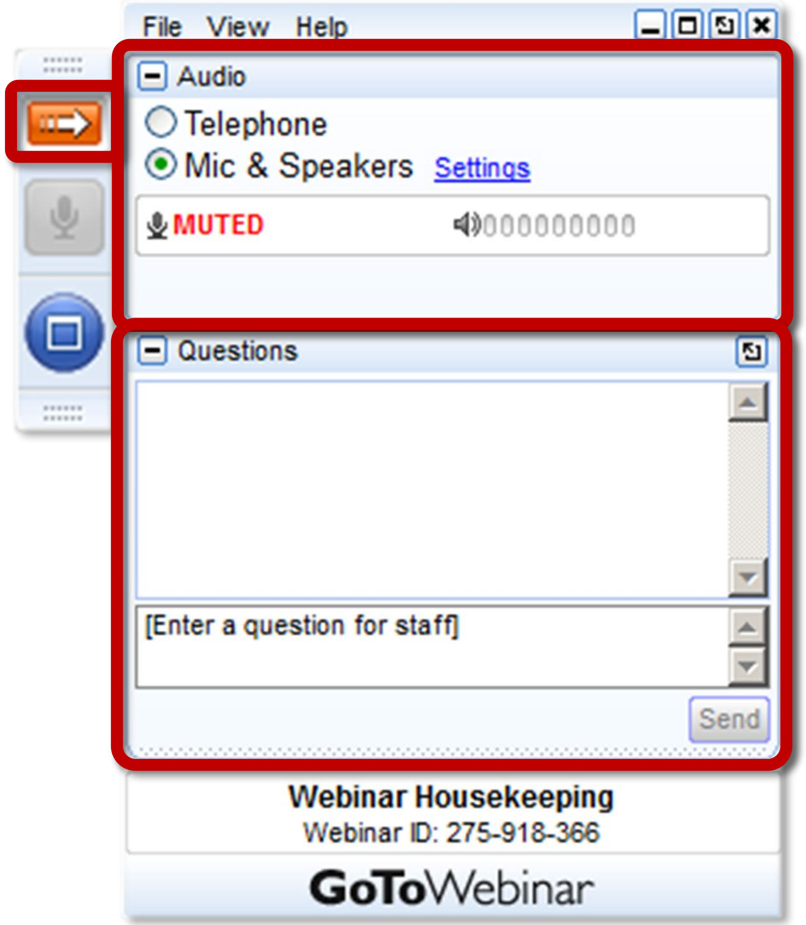
Slides presented during this webinar will be posted here:

- http://www.wirelessinnovation.org/webinars

Recorded Webinar will be available on the Forum's You Tube Channel:

- https://www.youtube.com/channel/UCYUeZvOuJTP27OzoKsyys0w

Email Lee.Pucker@wirelessinnovation.org if you need more information



Note on International Participation

Participants in this webinar are reminded that the Forum is an international organization, and they are prohibited from disclosing export restricted or controlled information during the course of this webinar.

Participants wishing additional information on this prohibition are referred to the Forum's Policy on Restricted and Controlled Information (SDRF Policy 009) available on the web.

WInnForum Standards™

WIRELESS INNOVATION FORUM

Today's Agenda

Today's Moderator:

- Ken Dingman, L3 Harris and Chair of the WInnForum's Software Defined Systems Committee

WInnForum Facilities Approach

- Eric Nicollet, Thales

Overview of Time Service Facility,

- Chuck Linn, L3Harris, and
- David Hagood, Cynosure

Time Service Usage in SOSA

- Round table discussion

WInnForum Facilities Principles

Eric Nicollet (Thales)

Presentation to SOSA Mission Operation Subcommittee
8 November 2022

Agenda

Introduction

Principles for WInnForum Facility Standards

WInnForum Facility PSMs Mapping Rules

Introduction

WInnForum developed a formally structured framework for Facilities specification

Two technical report capture this structure

- TR-2007 Principles for WInnForum Facility Standards
- TR-2008 WInnForum Facility PSMs Mapping Rules

Consistent with 2 Facility finalized WInnForum Facilities

- TS-0008 Transceiver Facility V2.1
- TS-3004 Time Service Facility V1.1

WInnForum Standards®

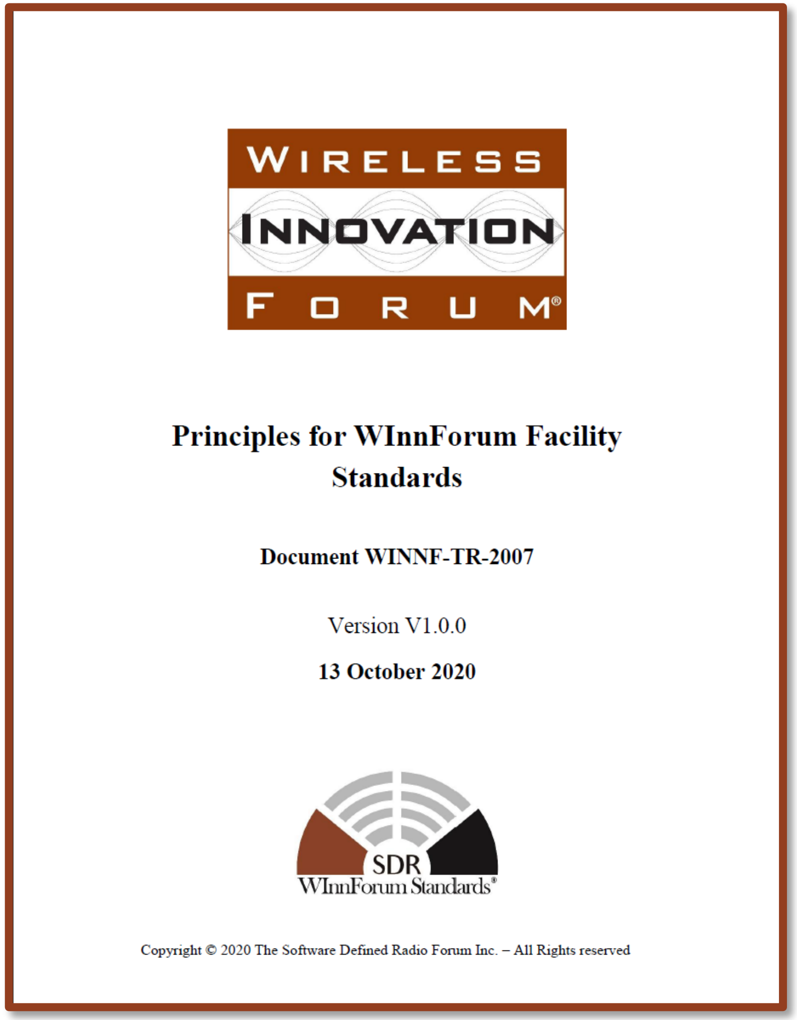
Principles for WInnForum Facility Standards

WINNF-TR-2007-V1.0.0

- October 2020
- 13 pages

Technical Report

Sets the reference concepts for specification of WInnForum Facilities


WIRELESS INNOVATION FORUM

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

Introduction (§ 1)

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

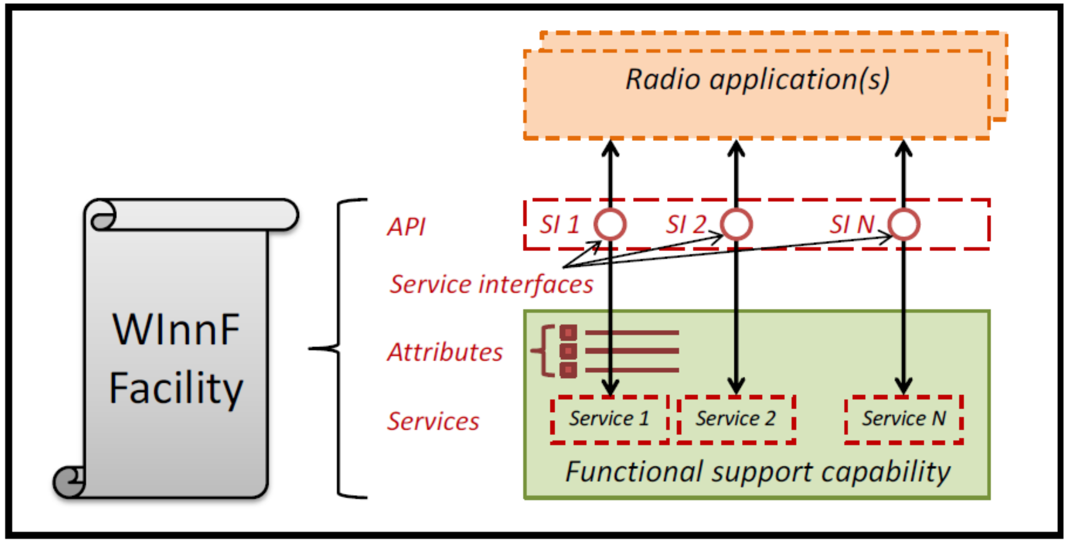


Figure 1 WInnForum facility overview

➔ **What is a WInnForum Facility?**

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

WInnForum Standards

WIRELESS INNOVATION FORUM

General principles (§ 2)

Radio capability

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).

Software Defined Radio and radio applications

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*.

Portability and hospitality

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*.

➔ **WInnForum Facilities aim at improving *portability* and *hospitality***

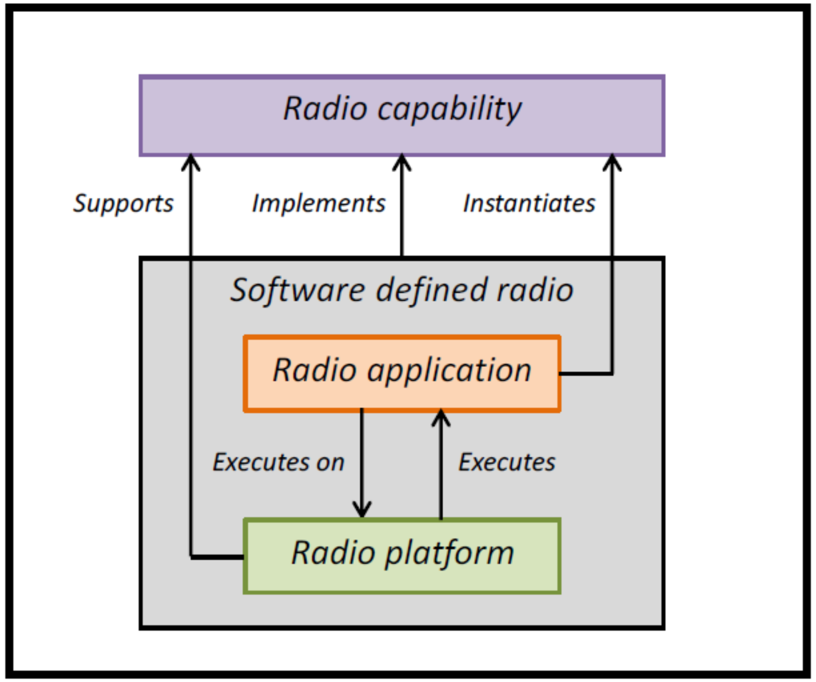


Figure 2 Base concepts

WInnForum Standards

WIRELESS INNOVATION FORUM

Architecture concepts (§ 3.1)

Component-based radio applications

Application component and processing nodes

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*.

➔ **Need to address a large variety of processing nodes: GPP, DSP, FPGA…**

WInnForum Standards®

WIRELESS INNOVATION FORUM®

Software support (§ 3.1.2)

Software support and environment

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*.

Some key constituents

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

➔ Not what WInnForum Facilities address

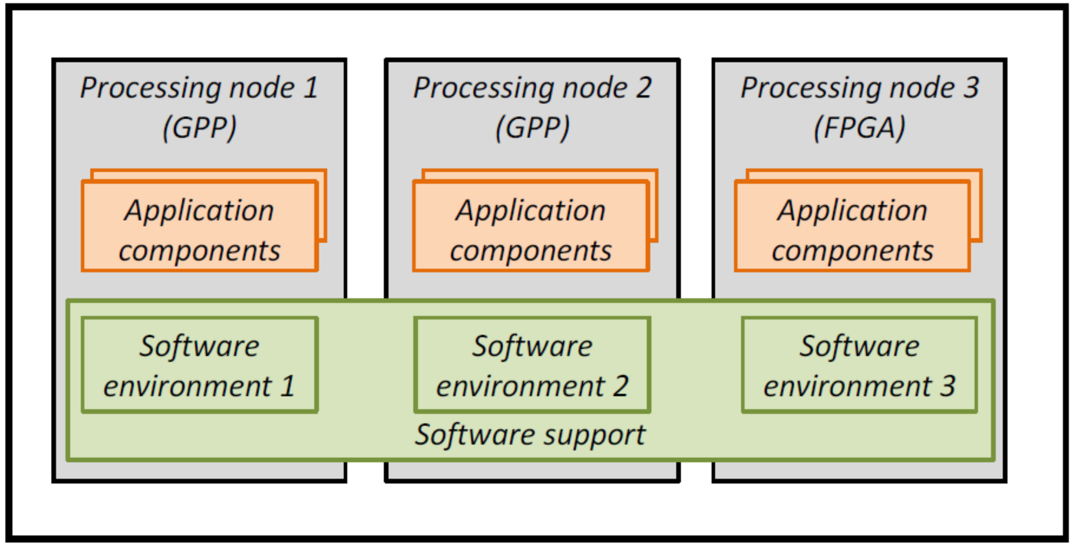


Figure 3 Software support

Functional support (§ 3.1.3)

Functional support and capabilities

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*.

Examples of Functional Support Capabilities

- Transceiver
- Time service
- Audio

Facades and access paradigms

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*.

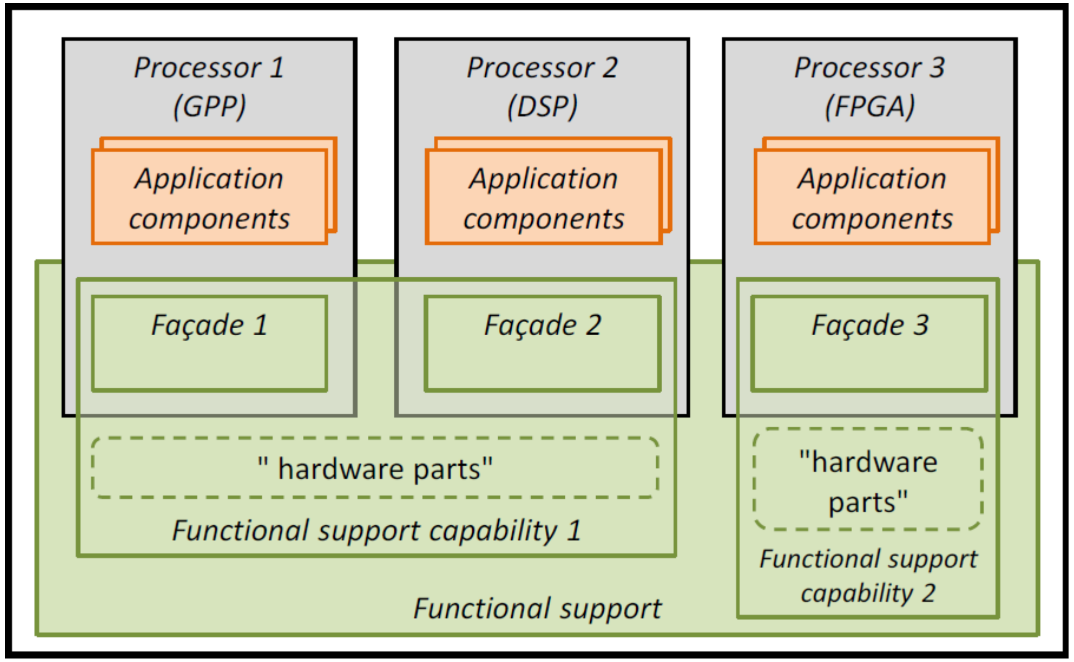


Figure 4 Functional support

➔ What WInnForum facilities adddress

Service-oriented functional support (§ 3.1.2)

Service and service name

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*.

Service implementation and interface

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.

1 service = 1 interface

- Very strong structural assumption

➔ WInnForum facilities not only specify the software interface of services, but the associated capability

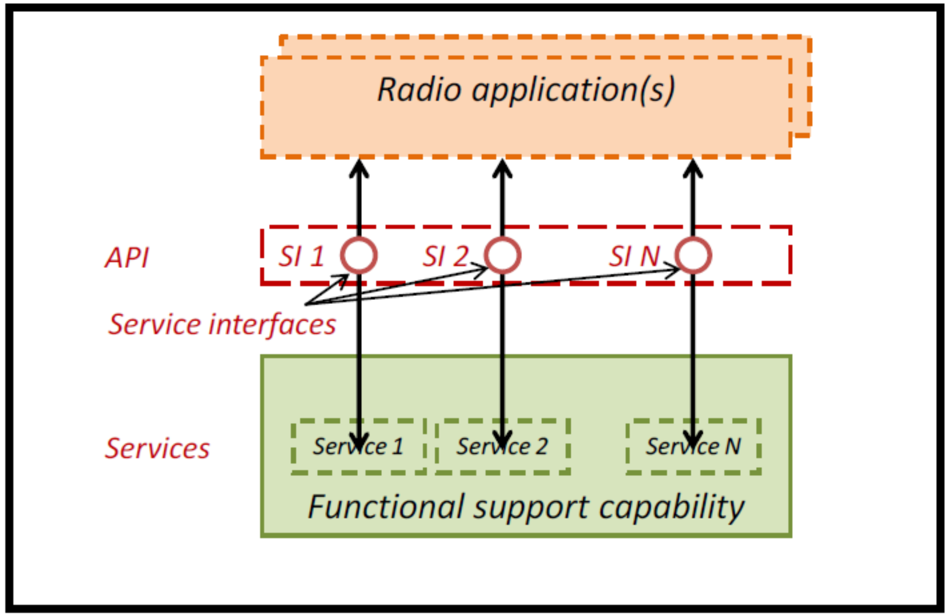


Figure 5 Services

WInnForum Standards®

WIRELESS INNOVATION FORUM

Provide and use services

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*.

Services groups

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*.

Primitive and implementation

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*.

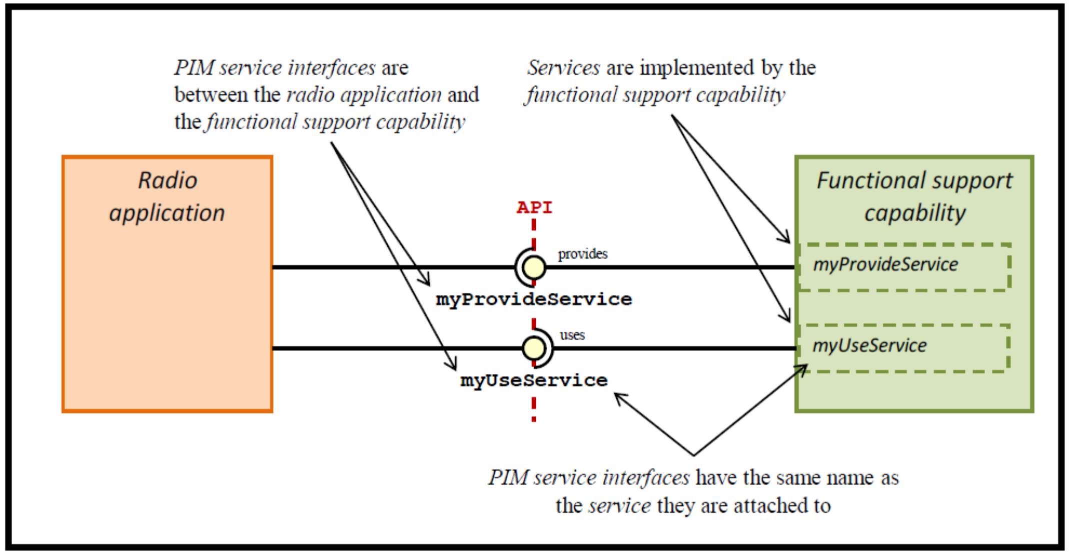


Figure 6 Services orientation

Real-time concepts (§ 3.2.5)

Call time and return time

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*.

Worst-case execution time

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}.

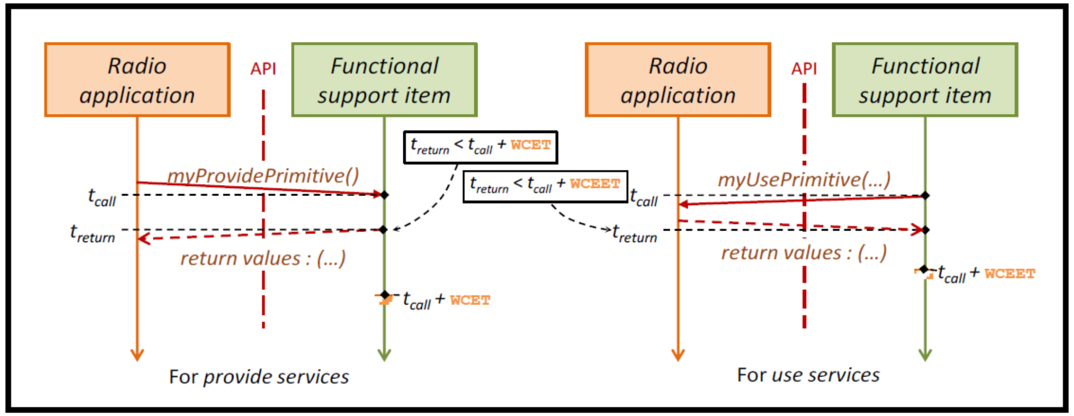


Figure 7 Services primitives call and return time

WInnForum Standards®

Facility attributes (§ 3.3)

Facility attributes

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

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

Counter-examples

- SWaP of implementations
- Features with no impact on radio application

Capabilities, properties and variables

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.

WInnForum Standards®

Specification principles (§ 4)

Facility composition

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).".

WInnForum Standards

Technical Report "WInnForum Facility PSMs Mapping Rules"

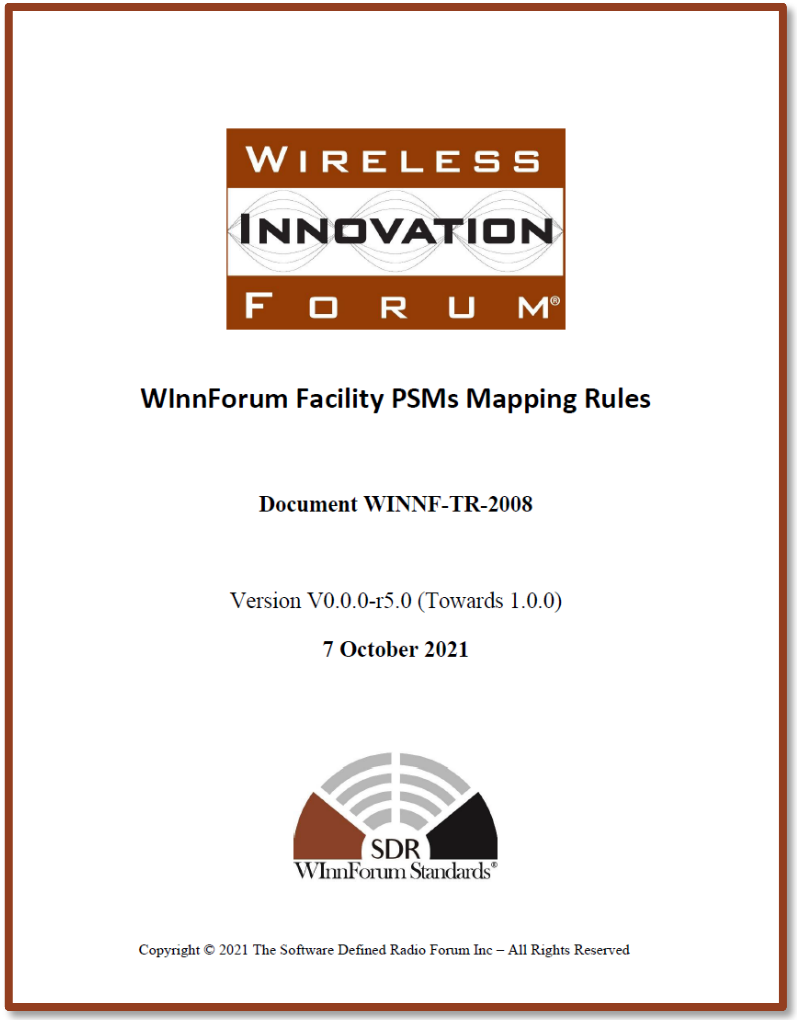
Mapping rules for programming paradigms

- Native C++
- SCA
- FPGA

WINNF-TR-2008 V1.0

Early 2022 (in approval)

42 pages


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

Reference architectural pattern (§ 1.2)

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*.

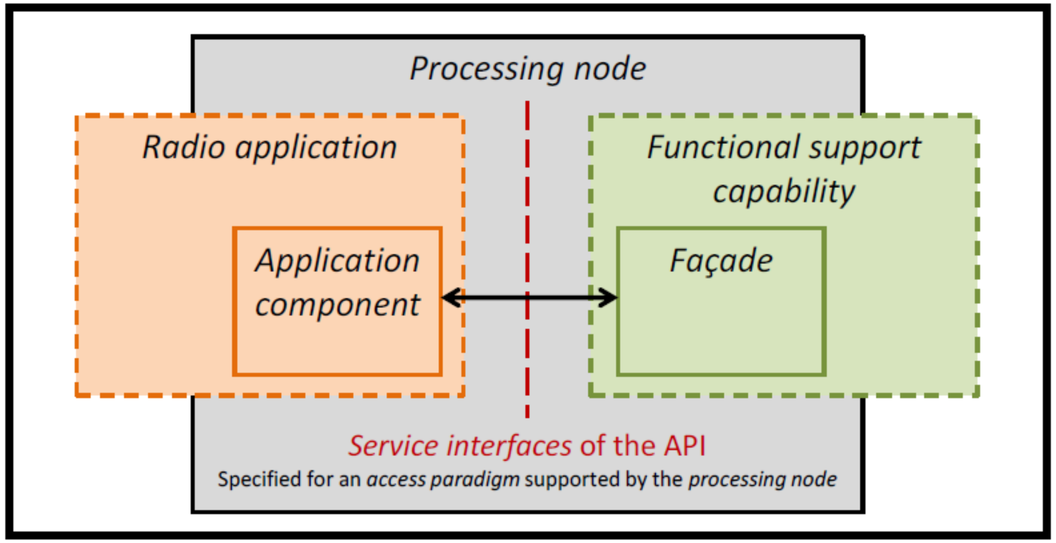


Figure 1 Reference architectural pattern

Native C++ - Overview (§ 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.

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*.

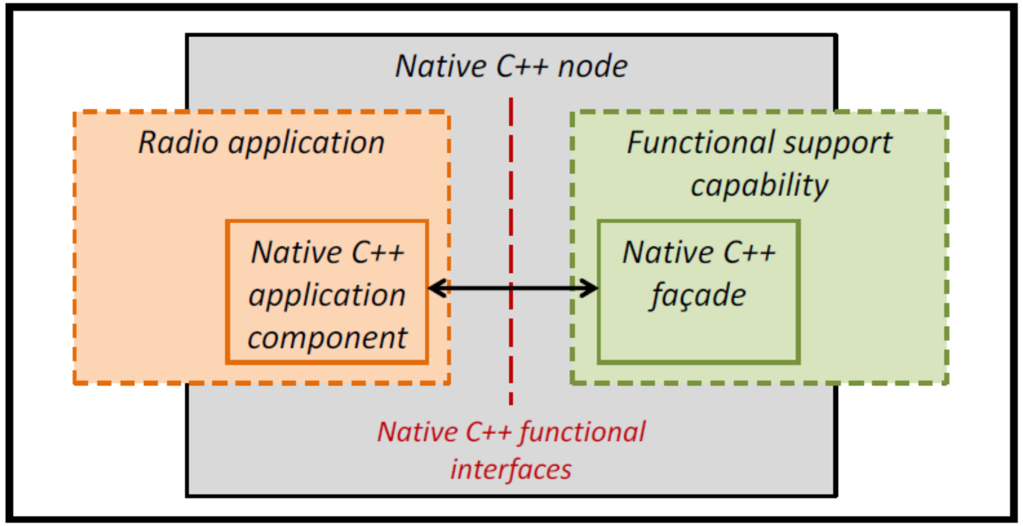


Figure 2 Positioning of *native C++ functional interfaces*

WInnForum Standards®

WIRELESS INNOVATION FORUM

Native C++ - 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).

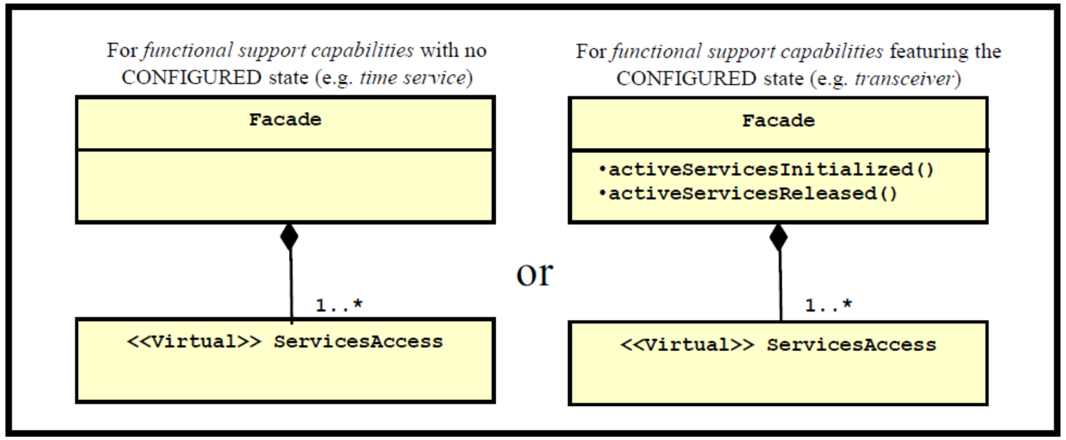


Figure 3 Class diagram of a *native C++ façade*

WInnForum Standards®

WIRELESS INNOVATION FORUM

Natice C++ - Explicit or generic services access

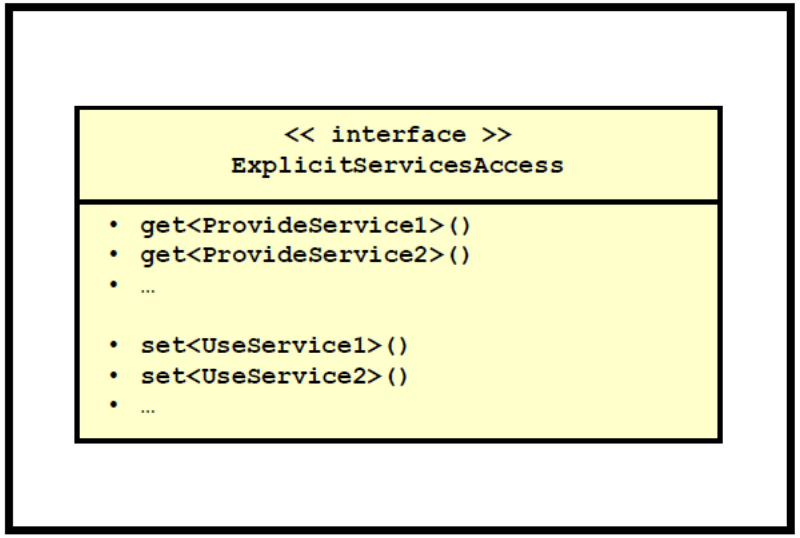


Figure 4 Class diagram of *explicit services access*

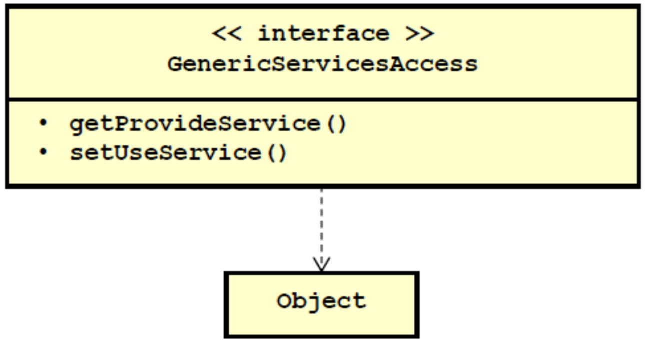


Figure 5 Class diagram of *generic services access*

SCA – Overview (§ 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*.

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*.

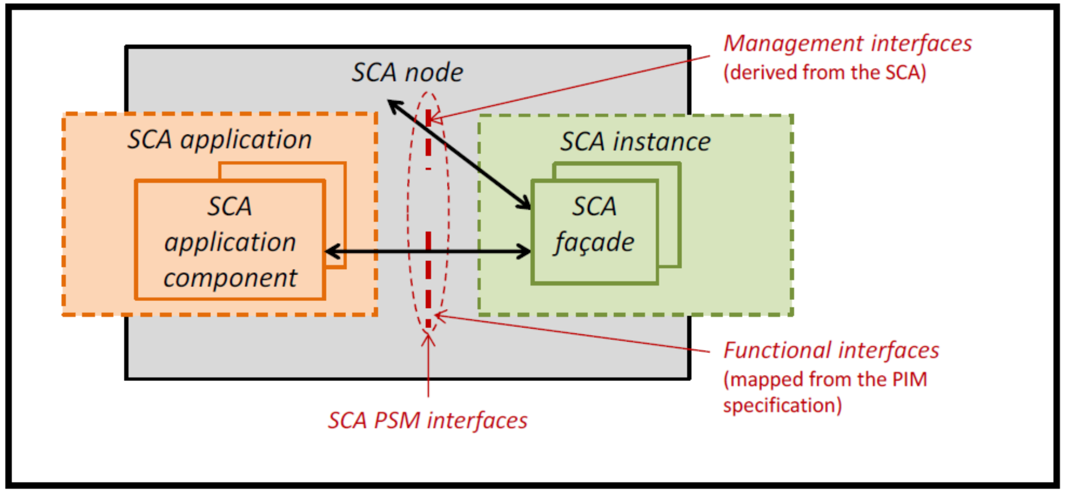


Figure 6 Architecture concepts for SCA PSMs

WInnForum Standards®

WIRELESS INNOVATION FORUM

SCA 2.2.2 Management Interfaces

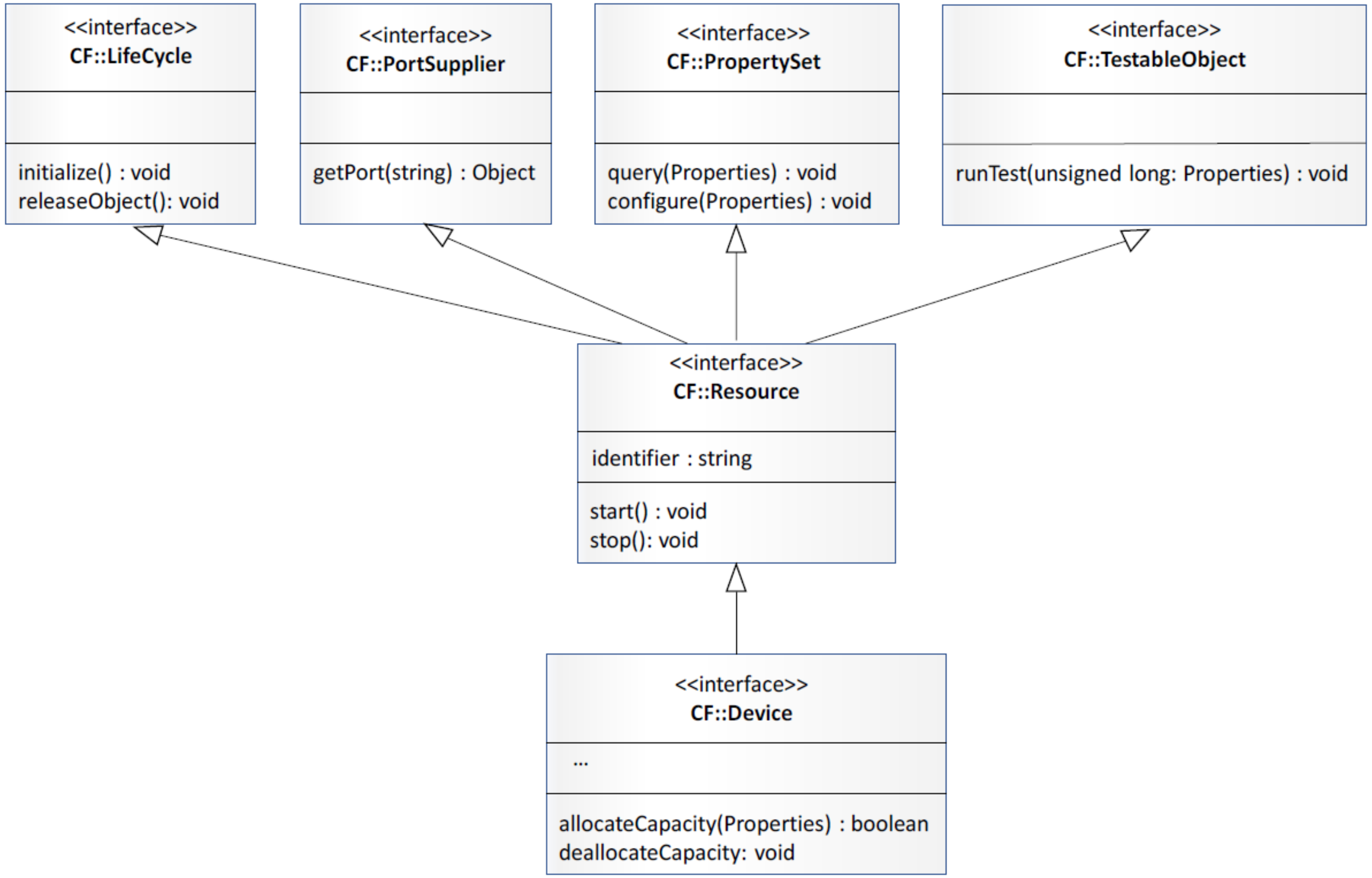


Figure 7 ***SCA 2.2.2 PSM management interfaces***

WInnForum Standards®

WIRELESS INNOVATION FORUM

SCA 4.1 Management Interfaces

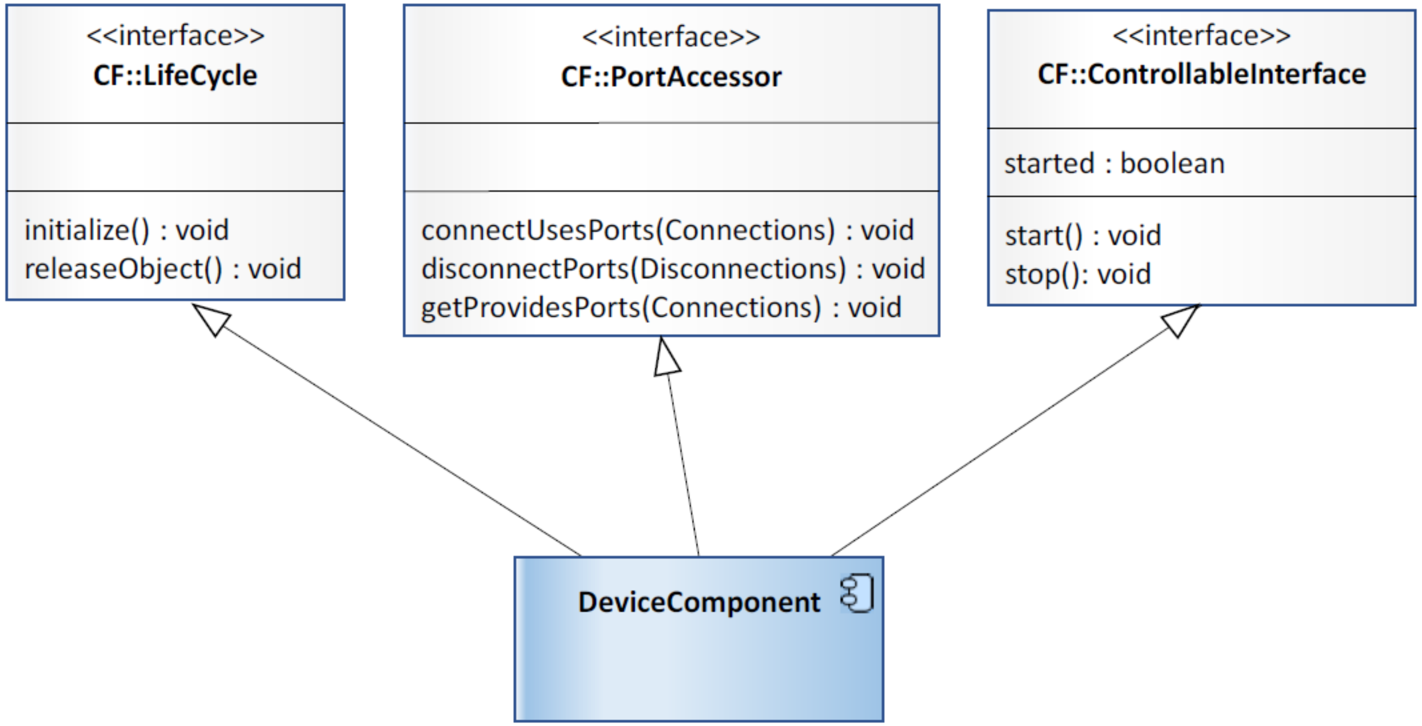


Figure 8 ***SCA 4.1 PSM management interfaces***

WInnForum Standards®

WIRELESS INNOVATION FORUM

FPGA – Overview (§ 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*.

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.

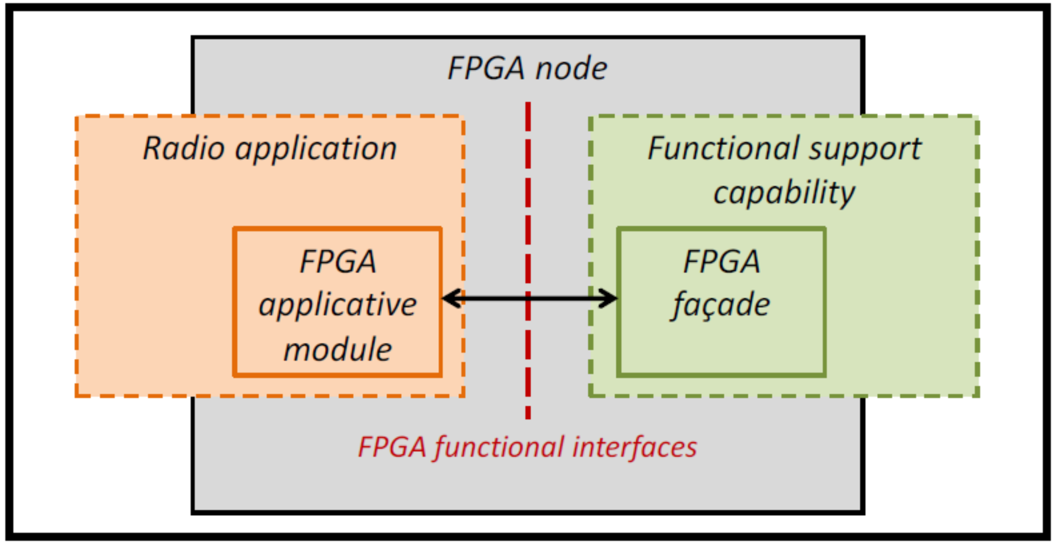


Figure 9 Positioning of *FPGA functional interfaces*

WInnForum Standards

WIRELESS INNOVATION FORUM

FPGA – Reference RTL signals

Structural 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*

Paramaters RTL signals

RTL signal name <FSC_TAG>_<instNum>_<PRIM_NAME>_+	Origin	Format	Usage case	Specification
EN_IN	*Caller*	1-bit signal	in param(s) or explicit return.	The *FPGA primitive* 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*

Semantics 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*

End of the presentation
Thank you for your attention

Any questions?

Contact:

eric.nicollet@thalesgroup.com

WInnForum Time Service Facility

David Hagood (Cynosure)
Chuck Linn (L3Harris Corporation)

Presentation to SOSA Mission Operation Subcommittee
8 November 2022

Agenda

Introduction

The PIM (Platform Independent Model) Specification

The PSM (Platform Specific Model) Specifications

Integration into SOSA

WInnForum Standards®

WIRELESS INNOVATION FORUM

Why WInnForum Time Service Facility

Portability, re-usability and interoperability are the goals

- Time is a simple concept that is complicated in actuality
- Waveforms and applications need to deal with that complexity in a standardized way, across a broad range of environments

Range of platform topologies

- Broad range of processing topologies
- Broad range of processing element types
- Broad range of language and communications environments

Software-defined radios use time differently, have special needs

- Since radios have cross-node communication, their collective understanding of time can differ from a single platform's concept of time.
- We need a monotonic time source, but we also need an up-to-date time source
  - Time sources can change, TFOMS can change, updates needed
  - But hardware and software need a continuous time concept as well

WInnForum Time Service Facility v1.0

WInnForum TSF v1.0 is a WInnForum Facility (SDR standard)

- One PIM Spec: fully implementation agnostic API and Properties
  - Released: December 2020
- Several PSM Specs: PIM Spec mapping to programming paradigms: Native C++, FPGA and SCA
  - Released 18 January 2022
- Uses JTNC Timing Service as reference standard

A successful and international harmonization

- 9 contributors from 5 countries: Kereval, FKIE, L3Harris Corporation, Hitachi Kokusai Electric, JTNC, Leonardo, Thales, Viavi Solutions, Cynosure
- Reflecting SDR background from worldwide manufacturers
- Reflecting lessons learnt from
  - US programs and manufacturers developments
  - EUR military programs: ESSOR (ESP, FIN, FRA, ITA, POL, SWE) and SVFuA (GER)

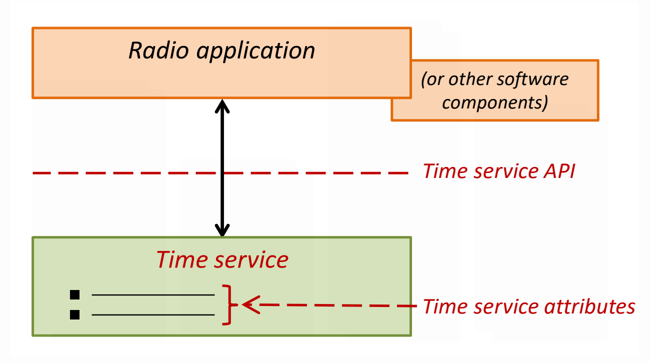
WInnForum Standards®

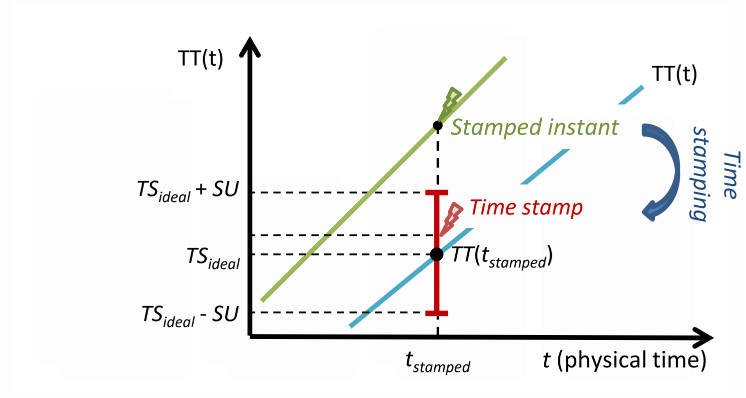
WIRELESS INNOVATION FORUM®

PIM and PSMs

Platform Independent Model defines the concepts supported.

- No specifics about language or implementation.
- Interfaces, semantics / behavior and flows are discussed
- The PIM forms the interoperable heart of the standard





Platform Specific Models define real implementations

- Native language implementations in C and C++
- Framework w. middleware (SCA / CORBA, DDS, etc. possible)
- FPGA implementations in VHDL

The PIM Specification

The Time Service Facility standardizes:

- The service-oriented Time Service Application Programming Interface (Time Service API)
- Standard associated Time Service Properties

A time service provides radio applications with

- Timing capabilities: providing radio applications with time
- Timer capabilities : triggering radio applications according to time-based conditions

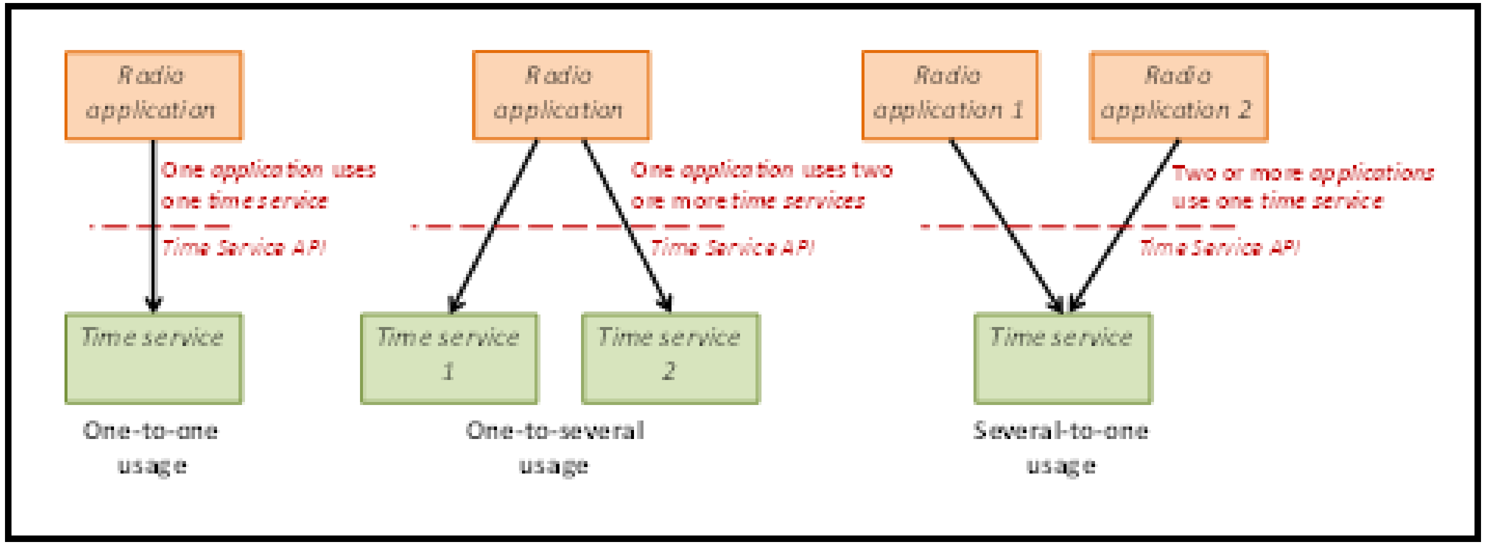
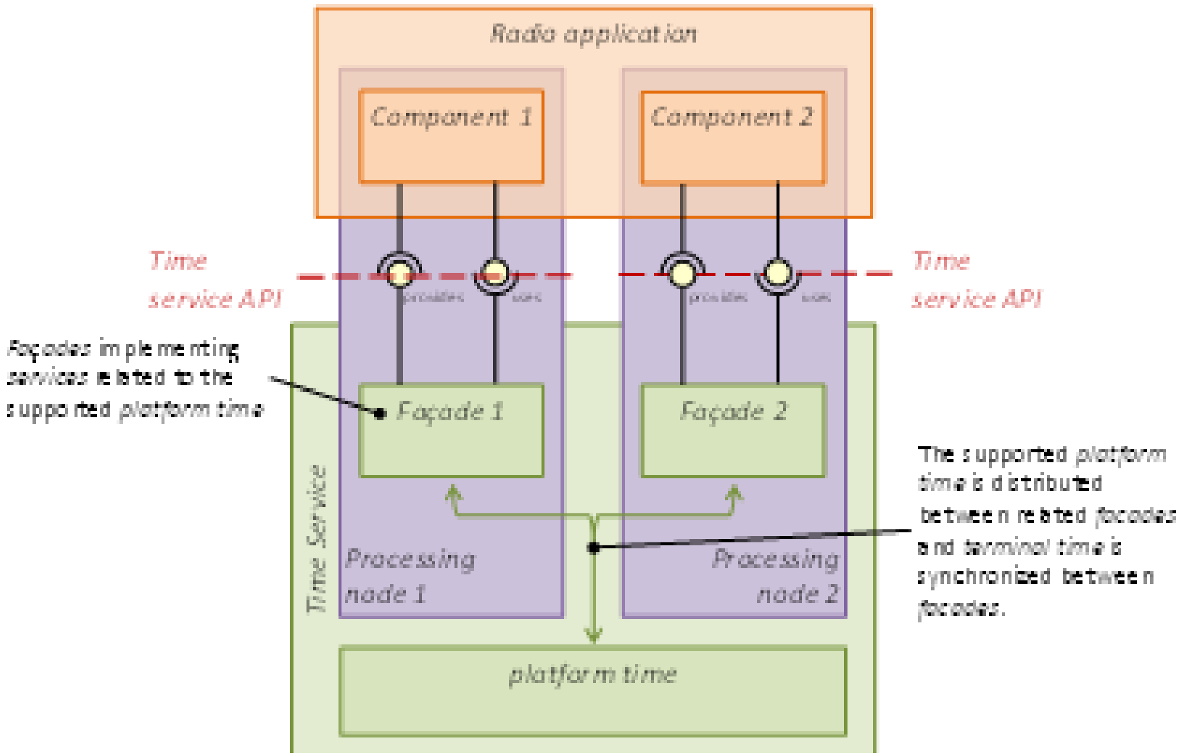


Figure 2 Possible time service usages

Distribution of coordinated time

- A *time service* is typically available on several processing elements of a system
- *Facades* are Individual access points of a *time service*, typically implemented on a per-processing element basis
- A *time service* coordinates 1+ *facades* with coordinated concepts of time
- Coordination is under responsibility of the *time service* implementation



WInnForum Standards®

WIRELESS INNOVATION FORUM

System Time

System time (ST) represents the radios best known estimation of the UTC (Coordinated Universal Time)

- Ideally, ST of all radios/nodes would correspond to the UTC
- In practice, each radio has a specific ST estimate of the UTC

The timing service may adjust ST at any moment

- Discontinuities in ST can happen in either direction
- Example: the inaccurate ST at radio boot will, once GNSS has synchronized, “jump” to a more accurate value, timing source handoffs, GPS denial, etc.

Radio application design is typically based on ST

- Example: TDMA slots of many waveforms are specified relative to the UTC, implying usage of the ST

Terminal Time

- ***Terminal time* (TT) provides the amount of time that has passed in the radio since an unspecified reference point**
- **TT is maintained using hardware oscillators of the radio**
- **TT is synchronized across the *façades* of a given *time service***
  - Not assumed to be synchronized across *time services* or *radios*
- **TT start time, epoch, or standard offset from ST is *unspecified***
  - TT is not equivalent of “looking at a clock” (that would be ST)
  - TT it is more about the platform hardware’s concept of time
  - <u>Example:</u> at radio boot, TT could be equal to 0 second
- **TT is monotonically increasing**
  - TT is never subject to changes, adjustments or rollover
  - TT increment period may be tweaked for oscillator disciplining

WInnForum Standards®

WIRELESS INNOVATION FORUM®

Typical usage of ST and TT

ST and TT are designed to work in tandem to enable a *radio application* to precisely place or tag platform events

Say a waveform *radio application*, having to begin a transmission at a given ST *start time* denoted ST_{start}

- E.g. ST_{start} = 0300.50 UTC

The platform start transmissions according to commands expressed in TT

- TT_{start} is time to be used to command transmission start

The waveform would therefore

- Use *getSystemTime()* to get an updated pair (ST_{ref} ;TT_{ref})
- Compute $TT_{start} = TT_{ref} + (ST_{start} – ST_{ref}) = ST_{start} – (ST_{ref} – TT_{ref})$
- Call the command primitive of the transceiver with TT_{start}

WInnForum Standards®

WIRELESS INNOVATION FORUM®

PIM services

Services groups (same as Modules)	Services (same as Interfaces)	Primitives	Optionality
`TerminalTime`	`TerminalTime::TerminalTimeAccess`	*getTerminalTime()*	MAN
`SystemTime`	`SystemTime::SystemTimeAccess`	*getSystemTime()* *getSystemTimeTfom()*	MAN
	`SystemTime::SystemTimeSetting`	*setSystemTime()*	OPT
`AppTime`	`AppTime::AppTimeHandling`	*pushAppTime()* *getAppTime()* *clearAppTime()*	OPT
`Timers`	`Timers:: [TBD]`		

Services groups (same as Modules)	Services (same as Interfaces)	Primitives	Optionality
`ExternalTimeRef`	`ExternalTimeRef::EtrEvent`	*pushEtrEvent()*	OPT

Essential SGs – TerminalTime

TerminalTime: 1 provide service

- `TerminalTimeAccess` fetches current Terminal Time value

SystemTime: 2 provided services

- `SystemTimeAccess` fetches the current *system time* estimate with a *terminal time* timeStamp and corresponding estimation error (mandatory)
- `SystemTimeSetting` to set the *system time (optional)*

AppTime: 3 provided services (optional)

- `AppTimeHandling` supports application time refs

ExternalTimeRef: 1 optional uses service (time serv → App)

- `EtrEvent` Pushes raw time data to app

WInnForum Standards®

Wireless Innovation Forum®

The PSM Specifications

PSM Specifications Overview

Planned PSM Specs

- Native C++
- FPGA
- SCA / CORBA

Possible other PSM Specs

- Native C
- Other IDL-based component frameworks, e.g. Redhawk
- SOSA / DDS, etc.. ?

User-specific PSMs will be possible, although discouraged from portability standpoint

Integration into SOSA

SOSA Time / Frequency services

Our understanding of time in SOSA is as follows:

- Time is provided to a card cage from a PNT card, using a stable clock (with radial distribution) and a 1 pps synchronization pulse
- The PNT card (if present) is responsible for selecting and transitioning between a stable oscillatory and external time sources The PNT card implements 6.7 module functions.
  - The PNT card could take external inputs or be replaced by an external clock and pps.
- Chassis PICs use backplane time + SDMs (?) to access time
- Equipment external to the chassis is provided with time using:
  - Reference clock + 1 pps, with NTP or MDM 1pps messages to provide “at the one pps, time will be” level information
  - -or- NTP + PTP alone (classic victory style)
- The 6.7 function will have APIs beyond backplane signals – but we don’t know the status of this work.

WInnForum Standards®

WIRELESS INNOVATION FORUM

SOSA Comms modality needs

Proposed SCA/SOSA interoperability approach for an SCA-based SDR PIC:

- SCA waveforms currently use either a JTNC time service or the WInnForum Time service facility (CORBA-based APIs) in conjunction with an underlying Terminal time distributed in hardware, synchronized.
- An SDR PIC would employ a Time Service with an implementation or shim to adapt the WInnForum APIs to the underlying SOSA standard time services will be needed inside the container
  - Depending on the native SOSA time support, this shim could range from trivial to complex.
- NOTE that in the case of the timing facility being included <u>only</u> in a comms PIC, that it would not be involved in selection or blending of the <u>best</u> time source, and would only affect elements within the comms PIC container.

Discussion questions

- **Character of SOSA “backplane” time:**
  - Is the backplane clock monotonically increasing, with no discontinuities?
  - When the Time/Freq PIC learns (say by acquiring GPS or connection of an external time source) of a “better” time which different from current time, what does it do in generating the system clock?
    - First, **is** the PIC the generator of the system clock?
    - Simply change the announcement at the next 1 pps mark (time discontinuity)
    - Slowly discipline the rate of the backplane clock until it converges with the new time concept. If so, is there a maximum rate delta?
- **Are there any usecases where a waveform would be used to serve as a time source into the SOSA system?**
- **Areas where the WInnForum Timing facility could help:**
  - Inside of Comms modality PICs
  - SOSA service as part of 6.7? Has API there yet been defined?
  - Reuse could be via a new PSM, or by cherry-picking concepts

WInnForum Standards®

WIRELESS INNOVATION FORUM®

End of the presentation
Thank you for your attention

Any questions?

Contacts:

- Chuck.Linn@L3Harris.com
- David.Hagood@cynoinc.com