Transceiver Facility SCA PSM specification

Transceiver Facility
where software defines the radio

Document WINNF-TS-0008-App02

Version V2.1.1

22 January 2022

TERMS, CONDITIONS & NOTICES

This document has been prepared by the work group of WInnF project SCA-2015-001 "Transceiver Next" to assist The Software Defined Radio Forum Inc. (or its successors or assigns, hereafter "the Forum"). It may be amended or withdrawn at a later time and it is not binding on any member of the Forum or of the Transceiver Next work group.

Contributors to this document that have submitted copyrighted materials (the Submission) to the Forum for use in this document retain copyright ownership of their original work, while at the same time granting the Forum a non-exclusive, irrevocable, worldwide, perpetual, royalty-free license under the Submitter's copyrights in the Submission to reproduce, distribute, publish, display, perform, and create derivative works of the Submission based on that original work for the purpose of developing this document under the Forum's own copyright.

Permission is granted to the Forum's participants to copy any portion of this document for legitimate purposes of the Forum. Copying for monetary gain or for other non-Forum related purposes is prohibited.

THIS DOCUMENT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY USE OF THIS SPECIFICATION SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE FORUM, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER, DIRECTLY OR INDIRECTLY, ARISING FROM THE USE OF THIS DOCUMENT.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the specification set forth in this document, and to provide supporting documentation.

This document was developed following the Forum's policy on restricted or controlled information (Policy 009) to ensure that that the document can be shared openly with other member organizations around the world. Additional Information on this policy can be found here: http://www.wirelessinnovation.org/page/Policies_and_Procedures

Although this document contains no restricted or controlled information, the specific implementation of concepts contain herein may be controlled under the laws of the country of origin for that implementation. Readers are encouraged, therefore, to consult with a cognizant authority prior to any further development.

Wireless Innovation Forum ™ and SDR Forum ™ are trademarks of the Software Defined Radio Forum Inc.

Table of Contents

List of Figures

List of Tables

Contributors

The following individuals and their organization of affiliation are credited as Contributors to development of the specification, for having been involved in the work group that developed the draft then approved by WInnF member organizations:

- Eric Nicollet, Thales,
- Claude Bélisle, VIAVI Solutions,
- David Hagood, Cynosure,
- Hugues Latour, VIAVI Solutions,
- François Lévesque, VIAVI Solutions,
- Steve Bernier, VIAVI Solutions,
- Marc Adrat, Fraunhofer FKIE,
- Guillaume Delbarre, DGA,
- Olivier Kirsch, KEREVAL,
- Charles Linn, L3Harris,
- David Murotake, HiKE,
- Kevin Richardson, MITRE.

Transceiver Facility SCA PSM specification

1 Introduction

This document WINNF-TS-0008-App02 is the *SCA PSM specification* of *Transceiver Facility* V2.1.0.

It derives from *Transceiver Facility PIM Specification* [Ref1] in accordance with *Principles for WInnForum Facility Standards* [Ref2].

It addresses the *Software Communications Architecture* (SCA) [Ref3] programming paradigm, applying the mapping rules of the SCA section of *WInnForum Facilities PSMs Mapping Rules* [Ref4] and specifically reporting any deviation to those rules.

The following figure positions of the interfaces addressed by the *SCA PSM specification*:

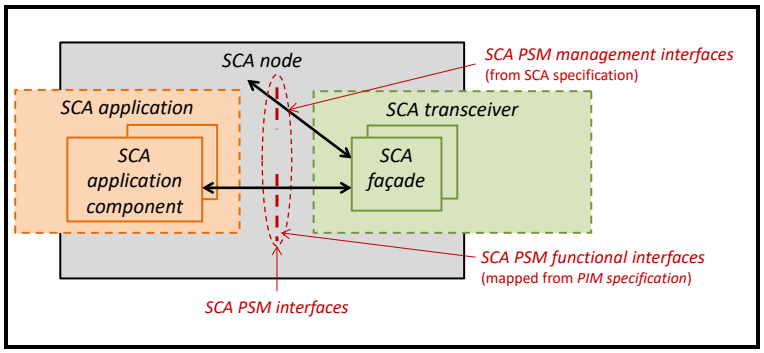

Figure 1 Interfaces addressed by *Transceiver Facility SCA PSM specification*

As depicted, the *SCA PSM specification* addresses the *SCA PSM management interfaces* and *SCA PSM functional interfaces* of *transceivers*, positioned, within an *SCA node*, between the *SCA application component*s of *SCA applications* and *SCA façades* of *SCA transceivers*.

It addresses version 2.2.2 [Ref5] and version 4.1 [Ref6] of the SCA.

It uses the abbreviation “xvcr” to identify *transceiver* in formal identifiers.

1.1 Reference definitions

The *Transceiver Facility SCA PSM specification* applies the following definitions from *Transceiver Facility PIM Specification* [Ref1]:

Topic	Used definitions
Transceiver technical concepts	*transceiver*, *Transceiver Facility*, *Tx channel*, *Rx channel*
Transceiver properties	*structure property*, *behavior property*, *notification property*, *interface declaration property*, *initialization property*, *storage property*, *parameter validity property*

Table 1 Definitions from *Transceiver Facility PIM Specification*

The *Transceiver Facility SCA PSM specification* applies the following definitions from *Principles for WInnForum Facility Standards* [Ref2]:

Topic	Used definitions
Base concepts	*radio application*
Architecture concepts	*façade*
WInnForum facility	*PIM specification*
Services	*service*, *service interface, provide service*, *use service*, *services group*
Primitives	*primitive*, *type*, *exception*
Attributes	*property*

Table 2 Definitions from *Principles for WInnForum Facility Standards*

The *Transceiver Facility SCA PSM specification* applies the following definitions from the SCA section of *WInnForum Facilities PSMs Mapping Rules* [Ref4]:

Topic	Used definitions
Specification purpose	*SCA PSM specification*, *SCA PSM functional interfaces, SCA PSM management interfaces*
Software architecture	*SCA node*, *SCA façade*, *SCA application*, *SCA application component*
SCA ports	*SCA functional port, service-wise assignment, services group-wise assignment*
SCA properties	*SCA PSM property*

Table 3 Definitions from SCA section of *WInnForum Facilities PSMs Mapping Rules*

The term "*unspecified"* indicates an aspect explicitly left to implementer's decisions.

1.2 Conformance

1.2.1 Radio platform items

An *SCA façade* of a *transceiver* implementation **is conformant with** the *Transceiver Facility SCA PSM specification* if it provides an SCA implementation of related *service interfaces*.

An *SCA transceiver* **is defined as** a *transceiver* implementation with all of its *SCA façades* being conformant with the *SCA PSM specification*.

1.2.2 Radio application items

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

1.3 SCA transceivers implementations

1.3.1 Localized or distributed

Depending on its architecture, an *SCA transceiver* can contain one or several *SCA façades*, as depicted in the following figure:

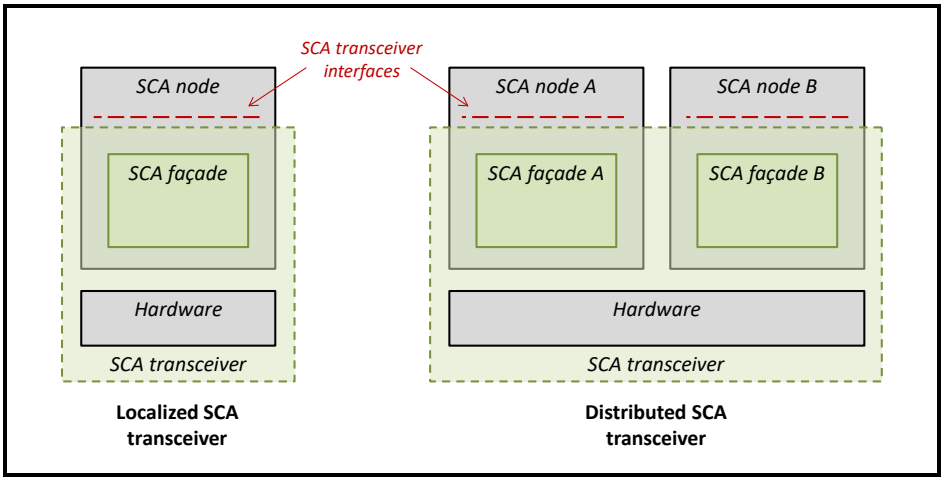

Figure 2 Localized or distributed SCA transceivers

1.3.2 Local or distant SCA connections

The following figure illustrates the concept of local or distant connections to an *SCA transceiver*:

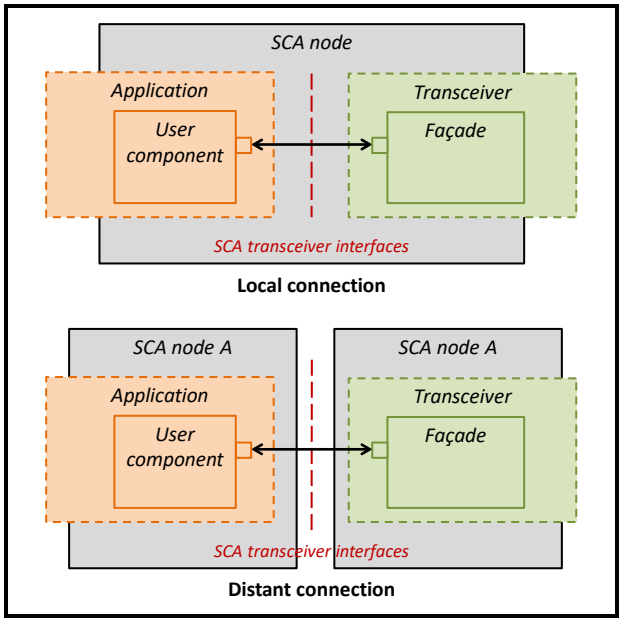

Figure 3 Concept of local or distant SCA connections

1.3.3 Hybrid SCA transceiver

The following figure illustrates the concept of hybrid *SCA transceiver*:

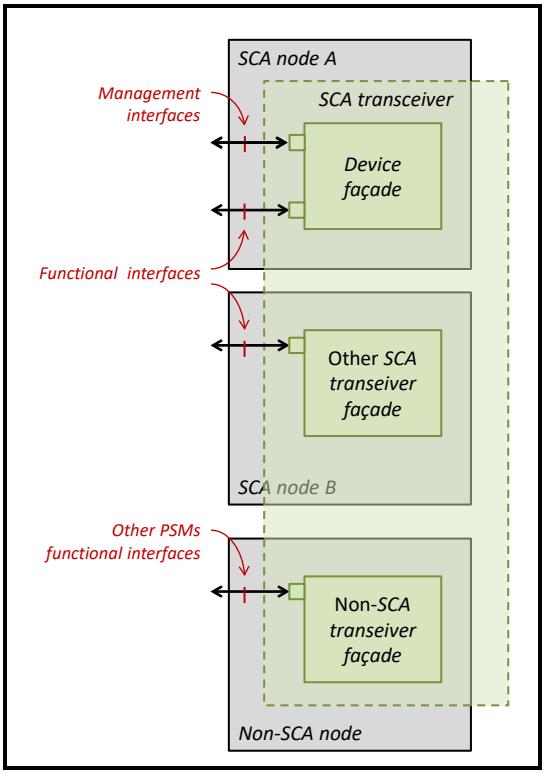

Figure 4 Concept of hybrid SCA transceiver

1.4 Document structure

Section 2 specifies the normative content for the *SCA PSM management interfaces*.

Section 3 specifies the normative classes for the *SCA PSM functional interfaces*.

Section 4 specifies the standard IDL files for the *SCA PSM functional interfaces*.

Section 5 specifies the *SCA PSM properties*.

2 SCA PSM management interfaces

The *SCA PSM management interfaces* specified in the SCA section of *WInnForum Facilities PSMs Mapping Rules* [Ref4] are applicable.

The *property* `XcvrScaVersion` (see section 5.2.2) indicates the used SCA version.

3 SCA PSM functional interfaces and ports

This normative section specifies the *SCA PSM functional interfaces* for *transceiver*, according to the SCA section of *WInnForum Facilities PSMs Mapping Rules* [Ref4].

3.1 Provide and Use Services Interfaces

The *service interfaces* for the *provide services* of the *SCA PSM functional interfaces* **are specified by** the following table:

PIM *service interface* (in `Transceiver::`)	PIM section	SCA PSM *service interface* (in `WInnF_Sca::Transceiver::`)
`Management::Reset`	2.4.1.1	`Management::Reset`
`Management::RadioSilence`	2.4.1.2	`Management::RadioSilence`
`BurstControl::DirectCreation`	2.4.2.1	`BurstControl::DirectCreation`
`BurstControl::RelativeCreation`	2.4.2.2	`BurstControl::RelativeCreation`
`BurstControl::AbsoluteCreation`	2.4.2.3	`BurstControl::AbsoluteCreation`
`BurstControl::StrobedCreation`	2.4.2.4	`BurstControl::StrobedCreation`
`BurstControl::Termination`	2.4.2.5	`BurstControl::Termination`

`BasebandSignal::SamplesTransmission`	2.4.3.2	`BasebandSignal16::SamplesExchange`
		`BasebandSignal32::SamplesExchange`
		`BasebandSignalFloat::SamplesExchange`

`BasebandSignal::RxPacketsLengthControl`	2.4.3.3	`BasebandSignal::RxPacketsLengthControl`
`Tuning::InitialTuning`	2.4.4.1	`Tuning::InitialTuning`
`Tuning::Retuning`	2.4.4.2	`Tuning::Retuning`
`GainControl::GainLocking`	2.4.6.2	`GainControl::GainLocking`
`TransceiverTime::TimeAcess`	2.4.7.1	`TransceiverTime::TimeAccess`
`Strobing::ApplicationStrobe`	2.4.8.1	`Strobing::ApplicationStrobe`

Table 4 Provide services functional services

The *service interfaces* for the *use services* of the *SCA PSM functional interfaces* **are specified by** the following table:

PIM *service interface* (in `Transceiver::`)	PIM section	SCA PSM *service interface* (in `WInnF_Sca::Transceiver::`)

`BasebandSignal::SamplesReception`	2.4.3.1	`BasebandSignal16::SamplesExchange`
		`BasebandSignal32::SamplesExchange`
		`BasebandSignalFloat::SamplesExchange`

`Notifications::Events`	2.4.5.1	`Notifications::Events`
`Notifications::Errors`	2.4.5.2	`Notifications::Errors`
`GainControl::GainChanges`	2.4.6.1	`GainControl::GainChanges`

Table 5 Use services functional interfaces

The PIM `BasebandSignal` module **maps to** a distinct PSM module for each type option: `BasebandSignal16` for 16-bit integer, `BasebandSignal32` for 32-bit integer and `BasebandSignalFloat` for 32-bit floating point.

The PIM `SamplesTransmission` and `SamplesReception` *service interfaces* **map to** a common PSM service interface `SamplesExchange`.

3.2 SCA PSM functional ports

This normative section specifies the *SCA functional ports* for the two possible assignment strategies.

3.2.1 Distribution between Tx and Rx channels

The *SCA functional ports* are distributed between *Tx channels* and *Rx channels* as illustrated in the following figure:

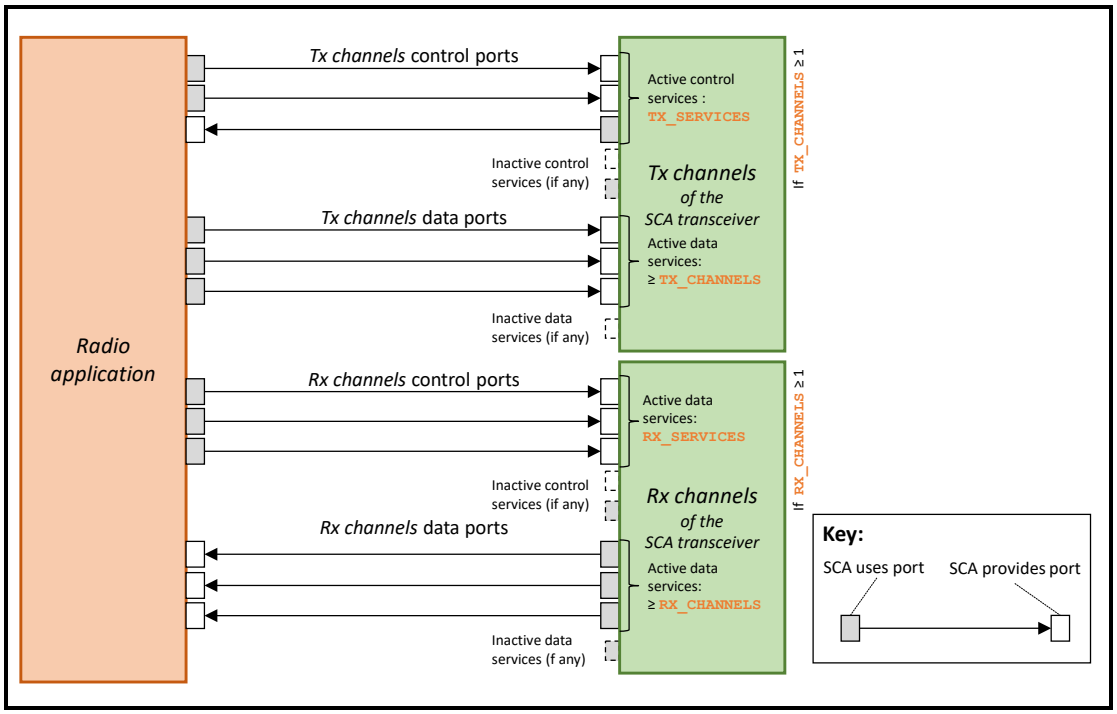

Figure 5 *SCA functional ports* distribution between *Tx channels* and *Rx channels*

3.2.1.1 Tx channels ports

An *SCA transceiver* implements *SCA functional ports* for *Tx channels* if one or more are active (*property* **TX_CHANNEL** > 0).

The *property* **TX_SERVICES** specify which *services* are attached to *Tx channels*.

The number of SCA provide ports implementing the **BasebandSignal<type>::SamplesExchange** interface is at least equal to **TX_CHANNELS**.

All other *SCA functional ports* for *Tx channels* are at most implemented once.

3.2.1.2 Rx channels ports

An *SCA transceiver* implements *SCA functional ports* for *Rx channels* if one or more are active (*property* **RX_CHANNEL** > 0).

The property **RX_SERVICES** specify which *services* are attached to *Rx channels*.

The number of *SCA* use ports implementing the **BasebandSignal<type>::SamplesExchange** interface is at least equal to **RX_CHANNELS**.

All other *SCA functional ports* for *Rx channels* are at most implemented once.

3.2.2 Service-wise assignment

The following figure illustrates the *SCA functional ports* for *Tx channels* with *service-wise assignment*:

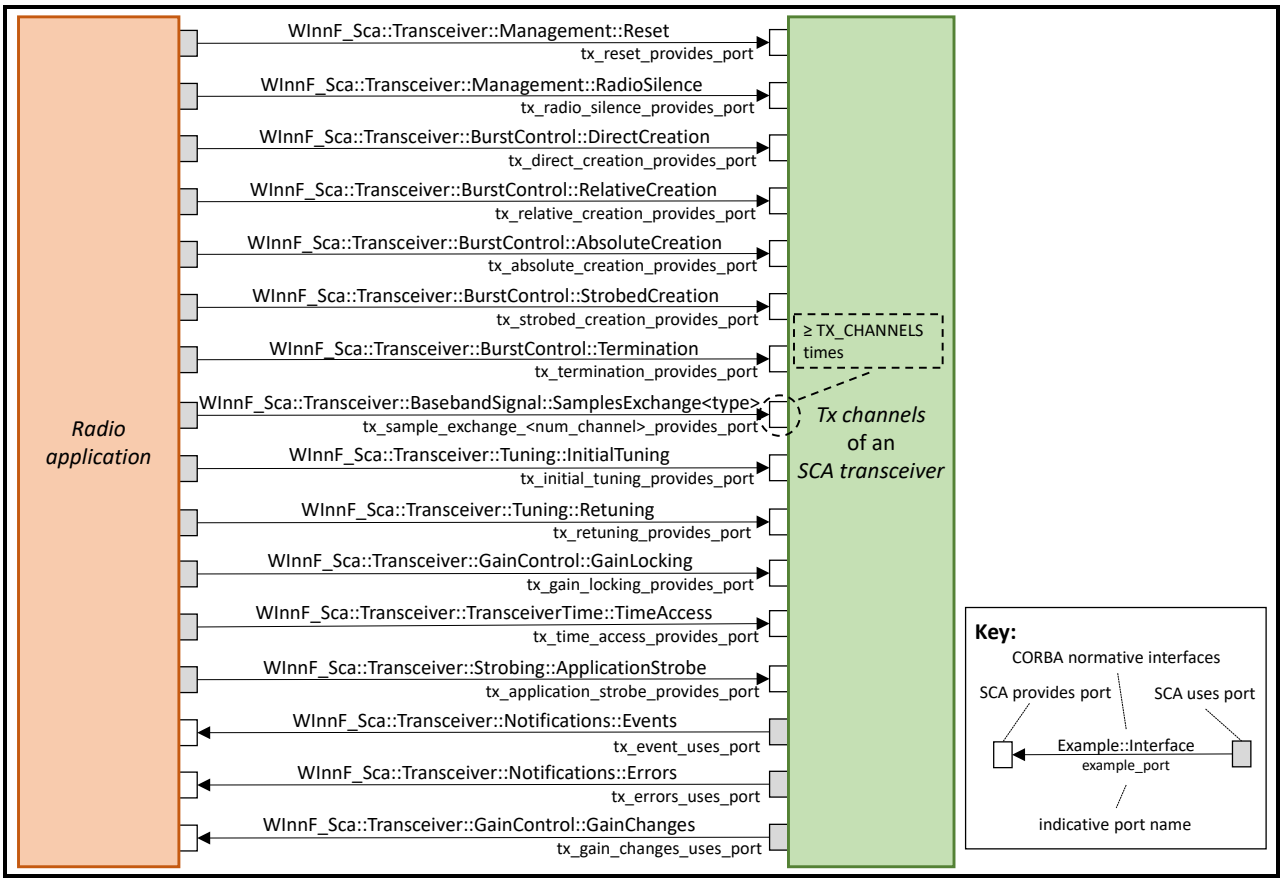

Figure 6 *SCA functional ports* for *Tx channels* with *service-wise assignment*

The following figure illustrates the *SCA functional ports* for *Rx channels* with *service-wise assignment*:

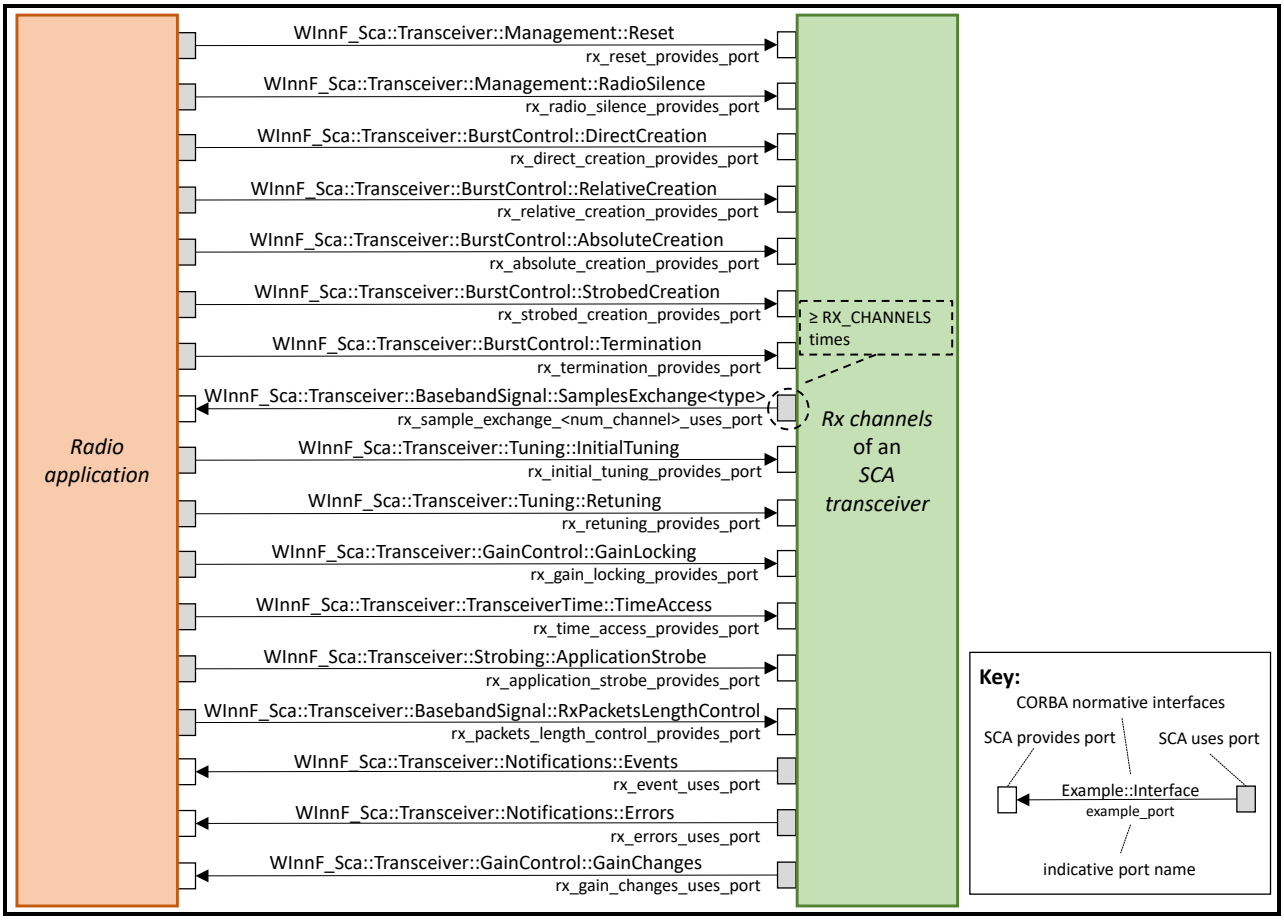

Figure 7 *SCA functional ports* for *Rx channels* with *service-wise assignment*

3.2.3 Services group-wise assignment

The following figure illustrates the *SCA functional ports* for *Tx channels* with *services group-wise assignment*:

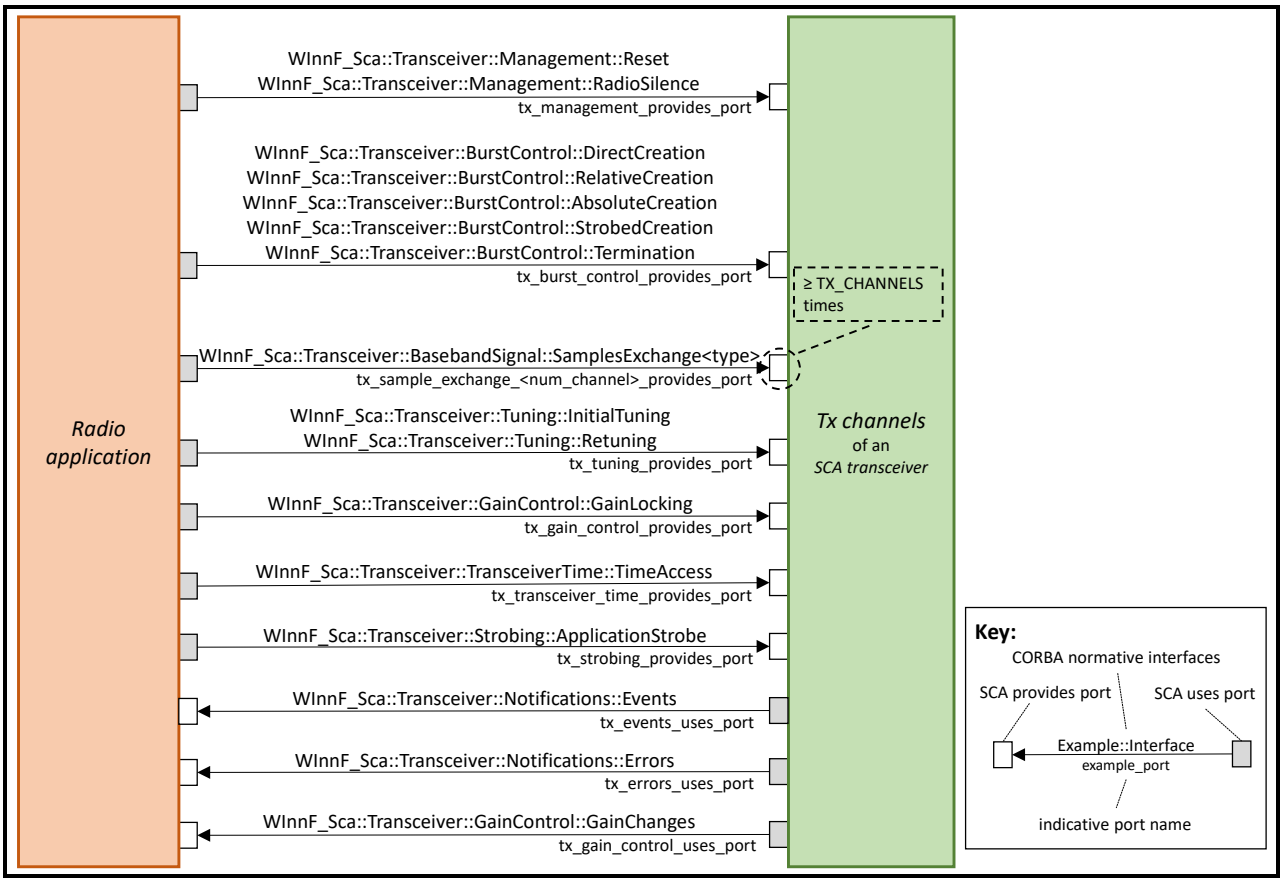

Figure 8 *SCA functional ports* for *Tx channels* with services group-wise assignment

The following figure illustrates the *SCA functional ports* for *Rx channels* with *services group-wise assignment*:

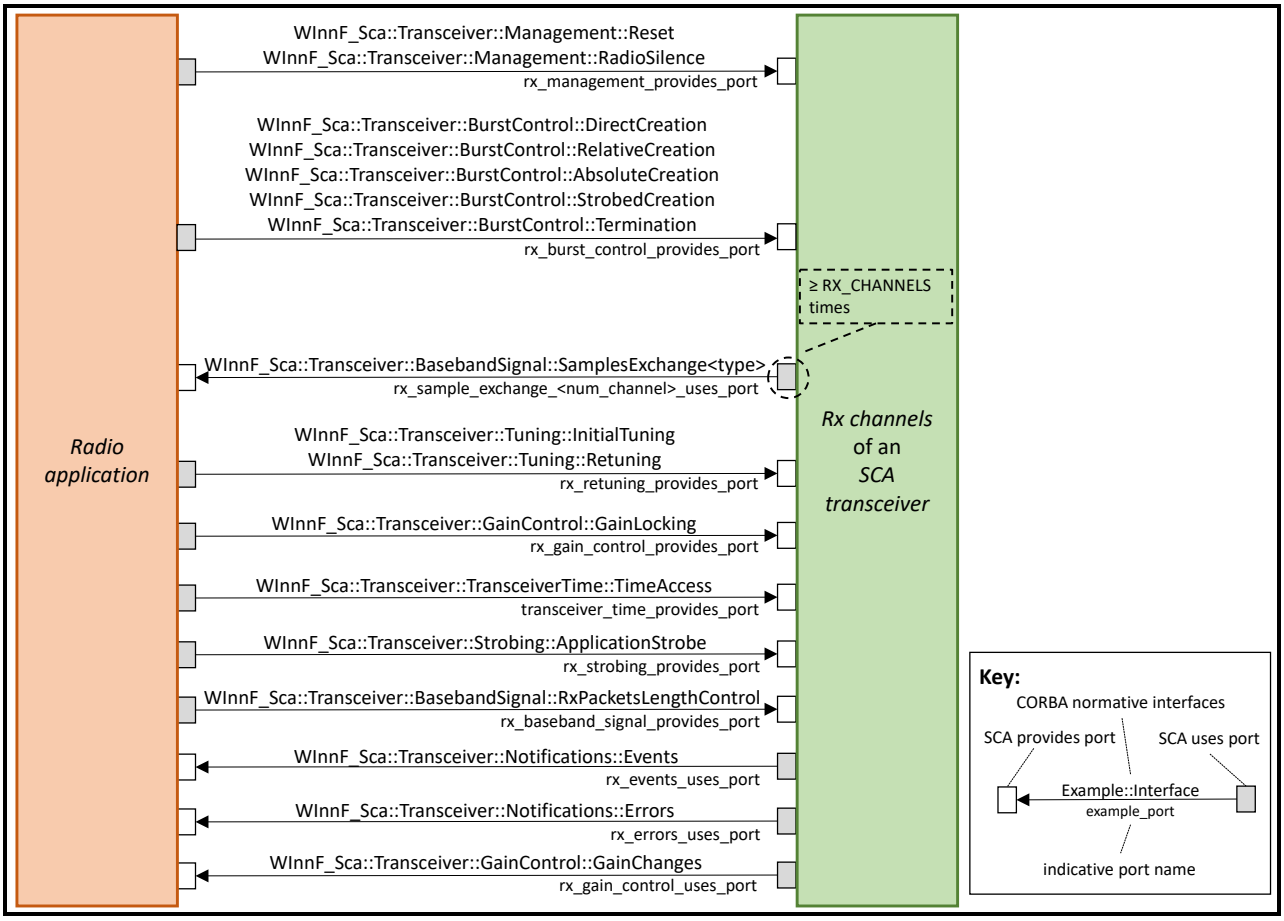

Figure 9 *SCA functional ports* for *Rx channels* with services group-wise assignment

3.2.4 Exceptions

For the *use service* `SamplesReception`, the *exceptions* `MaxPacketsLength` and `PacketsMILT`, although present in the IDL signature, are not to be raised by the *SCA application components*.

4 SCA PSM IDL files

This normative section specifies the standard IDL files (`.idl`) used by conformant *SCA transceiver* and *SCA application components* (see section 1.2).

The IDL files are processed by an IDL compiler to generate source code in a specific programming language from which the implementation is developed.

The specified IDL files have been successfully compiled using:

- TAO™ v2.5.0,
- ORBexpress® RT C++ v3.0.3.

4.1 Common

4.1.1 ScaXcvrTypes.idl

The `ScaXcvrTypes.idl` file **is specified as** the IDL file that declares the *types* specified by section 3.4 of *Transceiver Facility PIM Specification* [Ref1].

IDL support of the options addressed by *interface declaration properties* ([Ref1], section 4.5) implies to use different IDL files, with their names distinguishing the selected options, in order to avoid multiple definitions in the same IDL interface.

The number of supported options has therefore been limited to the greatest extent, while preserving options enabling significant savings in execution resources.

This resulted in the following choices:

- For `CARRIER_FREQ_TYPE` and `DELAY_TYPE`, the SCA PSM IDL **always uses** 64-bit types, since the related *types* appear in control primitives, for which the execution resources saving is low when taking lower size data *types*,
- For `IQ_TYPE`, the SCA PSM IDL **reflects the PIM types options** (16-bit, 32-bit or 32-bit floating point) in specifying one IDL file for each possibility, because of the execution resources savings at stake,
- For `TX_META_DATA` and `RX_META_DATA`, the SCA PSM IDL **always uses** parameters of type `CF::Properties` as specified by the SCA ([Ref5] and [Ref6]):
  - If `TX_META_DATA` or `RX_META_DATA` is `true`, meta-data are actually supported and the nature of the meta-data remains *unspecified*,
  - In case `TX_META_DATA` or `RX_META_DATA` is `false`, meta-data are not supported, therefore zero-length `CF::Properties` are sent over the interface.

The content of `ScaXcvrTypes.idl` **is specified as**:

```
#ifndef __SCA_XCVR_TYPES_DEFINED
#define __SCA_XCVR_TYPES_DEFINED

#include "ScaXcvrExceptions.idl"

module WInnF_Sca
{
  module Transceiver
  {
```

```
    //Gain
    typedef short Gain; // in 1/10 dB
    const Gain UNDEFINED_GAIN = 0x7FFF;

    // Delay
    typedef unsigned long long DelayType;
    typedef DelayType Delay;  // in ns

    const Delay UNDEFINED_DELAY = 0x7FFFFFFFFFFFFFFF;  // 9223372036854775807

    // BurstNumber
    typedef unsigned long BurstNumber;


    // TimeSpec
    struct TimeSpec
    {
      unsigned long seconds;             // in seconds
      unsigned long nanoseconds;         // in nanoseconds (<1.000.000.000)
    };

    // A TimeSpec with an undefined value shall have both, the seconds and
the
    // nanoseconds, fields set to UNDEFINED_TIME_SPEC_TIME
    const unsigned long UNDEFINED_TIME_SPEC_TIME = 0xFFFFFFFF;

  }; // Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_TYPES_DEFINED
```

4.1.2 ScaXcvrExceptions.idl

The **ScaXcvrExceptions.idl** file **is specified as** the IDL file that declares the *exceptions* specified by section 3.2 of the *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaTsfExceptions.idl** **is specified as**:

```
#ifndef __SCA_XCVR_EXCEPTIONS_DEFINED
#define __SCA_XCVR_EXCEPTIONS_DEFINED

module WInnF_Sca
{
  module Transceiver
  {
    exception NoAlternateReferencing
    {
    };

    exception NoOngoingProcessing
    {
    };

    exception UnavailableStrobeSource
    {
    };

    exception MinBlockLength
    {
    };

    exception MaxBlockLength
```

```
    {
    };

    exception MinCarrierFreq
    {
    };

    exception MaxCarrierFreq
    {
    };

    exception MinFromOngoing
    {
    };

    exception MaxFromOngoing
    {
    };

    exception MinFromPrevious
    {
    };

    exception MaxFromPrevious
    {
    };

    exception MinFromStrobe
    {
    };

    exception MaxFromStrobe
    {
    };

    exception MinGain
    {
    };

    exception MaxGain
    {
    };

    exception MaxNanoSeconds
    {
    };

    exception MaxRxPacketsLength
    {
    };

    exception MaxTuningPreset
    {
    };

    exception MaxPacketsLength
    {
    };

    exception AbsoluteMILT
    {
    };
```

```
    exception RelativeMILT
    {
    };

    exception RetuningMILT
    {
    };

    exception TuningMILT
    {
    };

    exception PacketsMILT
    {
    };

    exception UnavailableService
    {
    };
  };
};
#endif // __SCA_XCVR_EXCEPTIONS_DEFINED
```

4.2 Management *services group*

4.2.1 ScaXcvrManagementReset.idl

The **ScaXcvrManagementReset.idl** file **is specified as** the IDL file that declares the **Management Reset** *service interface* specified by section 3.1.1 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXvcrManagementReset.idl** **is specified as**:

```
#ifndef __SCA_XCVR_MANAGEMENT_RESET_DEFINED
#define __SCA_XCVR_MANAGEMENT_RESET_DEFINED

#include "ScaXcvrTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module Management
    {
      interface Reset
      {
        void resetChannels() // PIM name is "reset()",
                             // changed for compliance with TAO
          raises( UnavailableService );
      };
    };
  }; //Transceiver
}; // WInnF_Sca
#endif // __SCA_XCVR_MANAGEMENT_RESET_DEFINED
```

4.2.2 ScaXcvrManagementRadioSilence.idl

The **ScaXcvrManagementRadioSilence.idl** file **is specified as** the IDL file that declares the **Management RadioSilence** *service interface* specified by section 3.1.2 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrManagementRadioSilence.idl** **is specified as**:

```
#ifndef __SCA_XCVR_MANAGEMENT_RADIO_SILENCE_DEFINED
#define __SCA_XCVR_MANAGEMENT_RADIO_SILENCE_DEFINED

#include "ScaXcvrTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {

    module Management
    {
      interface RadioSilence
      {
        void startRadioSilence()
          raises( UnavailableService );

        void stopRadioSilence()
          raises( UnavailableService );
      };
    };

  }; //Transceiver
}; // WInnF_Sca
#endif // __SCA_XCVR_MANAGEMENT_RADIO_SILENCE_DEFINED
```

4.3 BurstControl *services group*

4.3.1 ScaXcvrBurstControlTypes.idl

The **ScaXcvrBurstControlTypes.idl** file **is specified as** the IDL file that declares some of the *API types* used by the **BurstControl** services that are specified by section 3.4 of *Transceiver Facility PIM Specification* [Ref1] along with the exceptions related to the use of these types in the **BurstControl** services.

The content of **ScaXcvrBurstControlTypes.idl** **is specified as**:

```
#ifndef __SCA_XCVR_BURST_CONTROL_TYPES_DEFINED
#define __SCA_XCVR_BURST_CONTROL_TYPES_DEFINED

#include "ScaXcvrTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    // BlockLength
    typedef unsigned long BlockLength;
    const BlockLength UNDEFINED_BLOCK_LENGTH = 0xFFFFFFFF;
  }; //Transceiver
```

```
}; // WInnF_Sca

#endif // __SCA_XCVR_BURST_CONTROL_TYPES_DEFINED
```

4.3.2 ScaXcvrBurstControlDirectCreation.idl

The **ScaXcvrBurstControlDirectCreation.idl** file **is specified as** the IDL file that declares the **BurstControl DirectCreation** *service interface* specified by section 3.1.3 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrBurstControlDirectCreation.idl** **is specified as**:

```
#ifndef __SCA_XCVR_BURST_CONTROL_DIRECT_CREATION_DEFINED
#define __SCA_XCVR_BURST_CONTROL_DIRECT_CREATION_DEFINED

#include "ScaXcvrBurstControlTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module BurstControl
    {
      interface DirectCreation
      {
        void startBurst(in BlockLength requestedLength)
          raises( UnavailableService,
                  MinBlockLength,
                  MaxBlockLength);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BURST_CONTROL_DIRECT_CREATION_DEFINED
```

4.3.3 ScaXcvrBurstControlRelativeCreation.idl

The **ScaXcvrBurstControlRelativeCreation.idl** file **is specified as** the IDL file that declares the **BurstControl RelativeCreation** *service interface* specified by section 3.1.4 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrBurstControlRelativeCreation.idl** **is specified as**:

```
#ifndef __SCA_XCVR_BURST_CONTROL_RELATIVE_CREATION_DEFINED
#define __SCA_XCVR_BURST_CONTROL_RELATIVE_CREATION_DEFINED

#include "ScaXcvrBurstControlTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module BurstControl
    {
      interface RelativeCreation
      {
        void scheduleRelativeBurst(in boolean requestedAlternate,
                                   in Delay requestedDelay,
```

```
                                    in BlockLength requestedLength)
          raises( UnavailableService,
                 NoAlternateReferencing,
                 MinFromPrevious,
                 MaxFromPrevious,
                 MinBlockLength,
                 MaxBlockLength,
                 RelativeMILT);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BURST_CONTROL_RELATIVE_CREATION_DEFINED
```

4.3.4 ScaXcvrBurstControlAbsoluteCreation.idl

The **ScaXcvrBurstControlAbsoluteCreation.idl** file **is specified as** the IDL file that declares the **BurstControl AbsoluteCreation** *service interface* specified by section 3.1.5 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrBurstControlAbsoluteCreation.idl** **is specified as**:

```
#ifndef __SCA_XCVR_BURST_CONTROL_ABSOLUTE_CREATION_DEFINED
#define __SCA_XCVR_BURST_CONTROL_ABSOLUTE_CREATION_DEFINED

#include "ScaXcvrBurstControlTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module BurstControl
    {
      interface AbsoluteCreation
      {
        void scheduleAbsoluteBurst(in TimeSpec requestedStartTime,
                                   in BlockLength requestedLength)
          raises( UnavailableService,
                 MaxNanoSeconds,
                 MinBlockLength,
                 MaxBlockLength,
                 AbsoluteMILT);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BURST_CONTROL_ABSOLUTE_CREATION_DEFINED
```

4.3.5 ScaXcvrBurstControlStrobedCreation.idl

The **ScaXcvrBurstControlStrobedCreation.idl** file **is specified as** the IDL file that declares the **BurstControl StrobedCreation** *service interface* specified by section 3.1.6 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrBurstControlStrobedCreation.idl** **is specified as**:

```
#ifndef __SCA_XCVR_BURST_CONTROL_STROBED_CREATION_DEFINED
```

```
#define __SCA_XCVR_BURST_CONTROL_STROBED_CREATION_DEFINED

#include "ScaXcvrBurstControlTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    //Strobe
    typedef unsigned short StrobeSource;

    const StrobeSource APPLICATION_STROBE = 0;
    const StrobeSource TIME_REF_PPS = 1;
    const StrobeSource GNSS_PPS = 2;
    // Values ranging from 3 to 7 are reserved for User Defined Strobe.

    module BurstControl
    {
      interface StrobedCreation
      {
        void scheduleStrobedBurst(in StrobeSource requestedStrobeSource,
                                  in Delay requestedDelay,
                                  in BlockLength requestedLength)
          raises( UnavailableService,
                  UnavailableStrobeSource,
                  MinFromStrobe,
                  MaxFromStrobe,
                  MinBlockLength,
                  MaxBlockLength);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BURST_CONTROL_STROBED_CREATION_DEFINED
```

4.3.6 *ScaXcvrBurstControlTermination.idl*

The **`ScaXcvrBurstControlTermination.idl`** file **is specified as** the IDL file that declares the **`BurstControl Termination`** *service interface* specified by section 3.1.7 of *Transceiver Facility PIM Specification* [Ref1].

The content of **`ScaXcvrBurstControlTermination.idl`** **is specified as**:

```
#ifndef __SCA_XCVR_BURST_CONTROL_TERMINATION_DEFINED
#define __SCA_XCVR_BURST_CONTROL_TERMINATION_DEFINED

#include "ScaXcvrBurstControlTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module BurstControl
    {
      interface Termination
      {
        void setBlockLength(in BlockLength requestedLength)
          raises( UnavailableService,
                  NoOngoingProcessing,
                  MinBlockLength,
```

```
                MaxBlockLength);

        void stopBurst()
          raises(UnavailableService,
                 NoOngoingProcessing);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BURST_CONTROL_TERMINATION_DEFINED
```

4.4 BasebandSignal *services group*

4.4.1 ScaXcvrBasebandSignal16SamplesExchange.idl

The **ScaXcvrBasebandSignal16SamplesExchange.idl** file **is specified as** the IDL file that declares the **BasebandSignal16 SamplesExchange** to address *service interfaces* specified by section 3.1.8, and 3.1.9 of *Transceiver Facility PIM Specification* [Ref1], when **IQ_TYPE** is equal to *16bit*.

The content of **ScaXcvrBasebandSignal16SamplesExchange.idl** **is specified as**:

```
#ifndef __SCA_XCVR_BASEBAND_SIGNAL_16_SAMPLES_EXCHANGE_DEFINED
#define __SCA_XCVR_BASEBAND_SIGNAL_16_SAMPLES_EXCHANGE_DEFINED

#include "ScaXcvrTypes.idl"
#include "CFProperties.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module BasebandSignal16
    {
      typedef short IQ;

      struct BasebandSample
      {
        IQ valueI;
        IQ valueQ;
      };

      typedef sequence <BasebandSample> BasebandPacket;

      interface SamplesExchange
      {
        void pushPacket(in BasebandPacket packet,
                        in boolean endOfBlock,
                        in CF::Properties metaData)
          raises( UnavailableService,
                  MaxPacketsLength,
                  PacketsMILT);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BASEBAND_SIGNAL_16_SAMPLES_EXCHANGE_DEFINED
```

4.4.2 ScaXcvrBasebandSignal32SamplesExchange.idl

The **ScaXcvrBasebandSignal32SamplesExchange.idl** file **is specified as** the IDL file that declares the **BasebandSignal32 SamplesExchange** to address the *service interfaces* specified by section 3.1.8, and 3.1.9 of *Transceiver Facility PIM Specification* [Ref1], when **IQ_TYPE** is equal to *32bit*.

The content of **ScaXcvrBasebandSignal32SamplesExchange.idl** **is specified as**:

```
#ifndef __SCA_XCVR_BASEBAND_SIGNAL_32_SAMPLES_EXCHANGE_DEFINED
#define __SCA_XCVR_BASEBAND_SIGNAL_32_SAMPLES_EXCHANGE_DEFINED

#include "ScaXcvrTypes.idl"
#include "CFProperties.idl"

module WInnF_Sca
{
  module Transceiver
  {

    module BasebandSignal32
    {
      typedef long IQ;

      struct BasebandSample
      {
        IQ valueI;
        IQ valueQ;
      };

      typedef sequence <BasebandSample> BasebandPacket;

      interface SamplesExchange
      {
        void pushPacket(in BasebandPacket packet,
                        in boolean endOfBlock,
                        in CF::Properties metaData)
          raises( UnavailableService,
                  MaxPacketsLength,
                  PacketsMILT);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BASEBAND_SIGNAL_32_SAMPLES_EXCHANGE_DEFINED
```

4.4.3 ScaXcvrBasebandSignalFloatSamplesExchange.idl

The **ScaXcvrBasebandSignalFloatSamplesExchange.idl** file **is specified as** the IDL file that declares the **BasebandSignalFloat SampleExchange** to address the *service interfaces* specified by section 3.1.8, and 3.1.9 of *Transceiver Facility PIM Specification* [Ref1], when **IQ_TYPE** is equal to *floatingPoint*.

The content of **ScaXcvrBasebandSignalFloatSamplesExchange.idl** **is specified as**:

```
#ifndef __SCA_XCVR_BASEBAND_SIGNAL_FLOAT_SAMPLES_EXCHANGE_DEFINED
#define __SCA_XCVR_BASEBAND_SIGNAL_FLOAT_SAMPLES_EXCHANGE_DEFINED
```

```
#include "ScaXcvrTypes.idl"
#include "CFProperties.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module BasebandSignalFloat
    {
      typedef float IQ;

      struct BasebandSample
      {
        IQ valueI;
        IQ valueQ;
      };

      typedef sequence <BasebandSample> BasebandPacket;

      interface SamplesExchange
      {
        void pushPacket(in BasebandPacket packet,
                        in boolean endOfBlock,
                        in CF::Properties metaData)
          raises( UnavailableService,
                  MaxPacketsLength,
                  PacketsMILT);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BASEBAND_SIGNAL_FLOAT_SAMPLES_EXCHANGE_DEFINED
```

4.4.4 *ScaXcvrBasebandSignalRxPacketsLengthControl.idl*

The **`ScaXcvrBasebandSignalRxPacketsLengthControl.idl`** file **is specified as** the IDL file that declares the **`RxPacketsLengthControl`** *service interface* specified by section 3.1.10 of *Transceiver Facility PIM Specification* [Ref1].

The content of **`ScaXcvrBasebandSignalRxPacketsLengthControl.idl`** **is specified as**:

```
#ifndef __SCA_XCVR_BASEBAND_SIGNAL_RX_PACKETS_LENGTH_CONTROL_DEFINED
#define __SCA_XCVR_BASEBAND_SIGNAL_RX_PACKETS_LENGTH_CONTROL_DEFINED

#include "ScaXcvrTypes.idl"
#include "CFProperties.idl"

module WInnF_Sca
{
  module Transceiver
  {
    // RxPacketsLengthControl
    typedef unsigned long PacketLength;

    module BasebandSignal
    {
      interface RxPacketsLengthControl
      {
        void setRxPacketsLength(in PacketLength requestedLength)
          raises( UnavailableService,
```

```
                MaxRxPacketsLength);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_BASEBAND_SIGNAL_RX_PACKETS_LENGTH_CONTROL_DEFINED
```

4.5 Tuning *services group*

4.5.1 ScaXcvrTuningTypes.idl

The **ScaXcvrTuningTypes.idl** file **is specified as** the IDL file that declares some of the *API types* used by the **Tuning** services that are specified by section 3.4 of *Transceiver Facility PIM Specification* [Ref1] along with the exceptions related to the use of these types in the **Tuning** services.

The content of **ScaXcvrTuningTypes.idl** **is specified as**:

```
#ifndef __SCA_XCVR_TUNING_TYPES_DEFINED
#define __SCA_XCVR_TUNING_TYPES_DEFINED

#include "ScaXcvrTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    //CARRIER_FREQ_TYPE  is define as and IDL typedef using the typedef
convention
    typedef unsigned long long CarrierFreqType;

    typedef CarrierFreqType CarrierFreq; // in Hz

    const CarrierFreq UNDEFINED_CARRIER_FREQ = 0x7FFFFFFFFFFFFFFF;  //
9223372036854775807
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_TUNING_TYPES_DEFINED
```

4.5.2 ScaXcvrTuningInitialTuning.idl

The **ScaXcvrTuningInitialTuning.idl** file **is specified as** the IDL file that declares the **InitialTuning** *service interface* specified by sections 3.1.11 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrTuningInitialTuning.idl** **is specified as**:

```
#ifndef __SCA_XCVR_TUNING_INITIAL_TUNING_DEFINED
#define __SCA_XCVR_TUNING_INITIAL_TUNING_DEFINED

#include "ScaXcvrTuningTypes.idl"

module WInnF_Sca
{
  module Transceiver
```

```
  {
    //InitialTuning Types
    typedef unsigned short TuningPreset;
    const TuningPreset UNDEFINED_TUNING_PRESET = 0xFFFF;

    module Tuning
    {
      interface InitialTuning
      {
        void setTuning(in TuningPreset requestedPreset,
                       in CarrierFreq requestedFrequency,
                       in Gain requestedGain,
                       in BurstNumber scheduleAbsoluteBurst)
          raises( UnavailableService,
                  MaxTuningPreset,
                  MinCarrierFreq,
                  MaxCarrierFreq,
                  MinGain,
                  MaxGain,
                  TuningMILT);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_TUNING_INITIAL_TUNING_DEFINED
```

4.5.3 ScaXcvrTuningRetuning.idl

The **ScaXcvrTuningRetuning.idl** file **is specified as** the IDL file that declares the **Retuning** *service interface* specified by sections 3.1.12 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrTuningRetuning.idl** **is specified as**:

```
#ifndef __SCA_XCVR_TUNING_RETUNING_DEFINED
#define __SCA_XCVR_TUNING_RETUNING_DEFINED

#include "ScaXcvrTuningTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module Tuning
    {
      interface Retuning
      {
        void retune(in CarrierFreq requestedFrequency,
                    in Gain requestedGain,
                    in Delay requestedDelay)
          raises( UnavailableService,
                  NoOngoingProcessing,
                  MinCarrierFreq,
                  MaxCarrierFreq,
                  MinGain,
                  MaxGain,
                  MinFromOngoing,
                  MaxFromOngoing,
                  RetuningMILT);
      };
    };
  }; //Transceiver
```

```
}; // WInnF_Sca

#endif // __SCA_XCVR_TUNING_RETUNING_DEFINED
```

4.6 Notifications *services group*

4.6.1 ScaXcvrNotifications.idl

The **`ScaXcvrNotifications.idl`** file **is specified as** the IDL file that declares the **`Notifications`** *service interfaces* specified by sections 3.1.13 and 3.1.14 of *Transceiver Facility PIM Specification* [Ref1].

The content of **`ScaXcvrNotifications.idl`** **is specified as**:

```
#ifndef __SCA_XCVR_NOTIFICATIONS_DEFINED
#define __SCA_XCVR_NOTIFICATIONS_DEFINED

module WInnF_Sca
{
  module Transceiver
  {
    module Notifications
    {
      enum Event
      {
        PROCESSING_START_EVENT,
        PROCESSING_STOP_EVENT,
        SILENCE_START_EVENT,
        SILENCE_STOP_EVENT
      };

      enum Error
      {
        DELAYED_TUNING_ERROR,
        TUNING_TIMEOUT_ERROR,
        DELAYED_FIRST_SAMPLE_ERROR,
        FIRST_SAMPLE_TIMEOUT_ERROR,
        TRANSMISSION_UNDERFLOW_ERROR,
        RECEPTION_OVERFLOW_ERROR,
        SHORTER_TRANSMITTED_BLOCK_ERROR,
        LONGER_TRANSMITTED_BLOCK_ERROR
      };

      interface Events
      {
        void notifyEvent(in Event notifiedEvent);
      };

      interface Errors
      {
        void notifyError(in Error notifiedError);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_NOTIFICATIONS_DEFINED
```

4.7 GainControl *services group*

4.7.1 ScaXcvrGainControlGainLocking.idl

The **ScaXcvrGainControlGainLocking.idl** file **is specified as** the IDL file that declares the **GainControl GainLocking** *service interface* specified by section 3.1.16 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrGainControlGainLocking.idl** **is specified as**:

```
#ifndef __SCA_XCVR_GAIN_CONTROL_GAIN_LOCKING_DEFINED
#define __SCA_XCVR_GAIN_CONTROL_GAIN_LOCKING_DEFINED

#include "ScaXcvrTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module GainControl
    {
      interface GainLocking
      {
        void lockGain()
          raises( UnavailableService,
                  NoOngoingProcessing);

        void unlockGain()
          raises( UnavailableService,
                  NoOngoingProcessing);
      };
    };
  };  //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_GAIN_CONTROL_GAIN_LOCKING_DEFINED
```

4.7.2 ScaXcvrGainChanges.idl

The **ScaXcvrGainControlGainChanges.idl** file **is specified as** the IDL file that declares the **GainControl GainChanges** *service interface* specified by section 3.1.15 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrGainControlGainChanges.idl** **is specified as**:

```
#ifndef __SCA_XCVR_GAIN_CONTROL_GAIN_CHANGES_DEFINED
#define __SCA_XCVR_GAIN_CONTROL_GAIN_CHANGES_DEFINED

#include "ScaXcvrTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module GainControl
    {
      typedef unsigned long SampleNumber;

      interface GainChanges
```

```
      {
        void indicateGain(in Gain newGain,
                          in SampleNumber firstValidSample);
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_GAIN_CONTROL_GAIN_CHANGES_DEFINED
```

4.8 TimeAccess *services group*

4.8.1 ScaXcvrTransceiverTimeTimeAccess.idl

The **ScaXcvrTransceiverTimeTimeAccess.idl** file **is specified as** the IDL file that declares the **TransceiverTime TimeAccess** *service interface* specified by section 3.1.17 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrTransceiverTimeTimeAccess.idl** **is specified as**:

```
#ifndef __SCA_XCVR_TRANSCEIVER_TIME_TIME_ACCESS_DEFINED
#define __SCA_XCVR_TRANSCEIVER_TIME_TIME_ACCESS_DEFINED

#include "ScaXcvrTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module TransceiverTime
    {
      interface TimeAccess
      {
        void getCurrentTime(out TimeSpec currentTime)
          raises( UnavailableService );

        void getLastStartTime(out TimeSpec lastStartTime,
                              out BurstNumber lastBurstNumber)
          raises( UnavailableService );
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_TRANSCEIVER_TIME_TIME_ACCESS_DEFINED
```

4.9 ApplicationStrobe *services group*

4.9.1 ScaXcvrStrobingApplicationStrobe.idl

The **ScaXcvrStrobingApplicationStrobe.idl** file **is specified as** the IDL file that declares the **Strobing ApplicationStrobe** *service interface* specified by section 3.1.18 of *Transceiver Facility PIM Specification* [Ref1].

The content of **ScaXcvrStrobingApplicationStrobe.idl** **is specified as**:

```
#ifndef __SCA_XCVR_STROBING_APPLICATION_STROBE_DEFINED
```

```
#define __SCA_XCVR_STROBING_APPLICATION_STROBE_DEFINED

#include "ScaXcvrTypes.idl"

module WInnF_Sca
{
  module Transceiver
  {
    module Strobing
    {
      interface ApplicationStrobe
      {
        void triggerStrobe()
          raises( UnavailableService );
      };
    };
  }; //Transceiver
}; // WInnF_Sca

#endif // __SCA_XCVR_STROBING_APPLICATION_STROBE_DEFINED
```

5 SCA PSM Properties

This normative section specifies the *SCA PSM properties*.

5.1 Natures of SCA PSM properties

Three different natures of *SCA PSM properties* are considered:

- *SCA capability properties*,
- *SCA capacity properties*,
- *SCA configure properties*.

5.1.1 SCA capability properties

An *SCA capability property* **is defined as** an SCA property of kind "allocation" with an action type other than "external".

SCA capability properties enable to check that an *SCA transceiver* can support the needs of a *radio application*.

They are specified by the domain profiles (.prf) of both the *radio application* and the *radio platform*:

- The .prf of the *radio application* specifies the desired value,
- The .prf of the *SCA transceiver* specifies the list of possible values,
- For instantiation of a *radio application*, the Core Framework checks if the desired value belongs to the list of possible values. If not, the instantiation aborts.

5.1.2 SCA capacity properties

An *SCA capacity property* **is defined as** an SCA property of kind "allocation" with an action type equal to "external".

SCA capacity properties enable to manage consumption of resources by the instantiated *radio applications*:

- The .prf of the *radio application* specifies the desired amount,
- The .prf of the *SCA transceiver* indicates availability of the property and the total available amount,
- For instantiation of a *radio application*, if the property is available, the Core Framework reserves the amount desired by the *radio application* using *allocateCapacity()*. If the allocation fails, the instantiation aborts.

5.1.3 SCA configure properties

An *SCA configure property* **is defined as** an SCA property of kind "configure".

SCA configure properties enable to configure an *SCA transceiver* according to the needs of a *radio application*, before the `CONFIGURED` state is reached.

The Core Framework sets the *SCA configure properties* to the desired value using *configure()*.

5.2 Versions properties

5.2.1 PIM version

The `XcvrPimVersion` *SCA PSM property* **is specified as** an integer indicating the version of the *Transceiver Facility PIM Specification* [Ref1] from which the *SCA PSM specification* is derived.

The defined values of `XcvrPimVersion` **are specified as**:

PIM specification version	Value
V2.0.0	`0x020000`
V2.1.0	`0x020100`

Table 6 XcvrPimVersion defined values

`XcvrPimVersion` **is equal to** `0x020100`.

5.2.2 SCA version

The `XcvrScaVersion` *SCA PSM property* **is specified as** an integer indicating the used SCA version.

The defined values for `XcvrScaVersion` **are specified as**:

SCA version	Value
V2.2.2	`0x020202`
V4.1	`0x040100`

Table 7 XcvrScaVersion defined values

5.3 PIM-derived properties

The PIM-derived *SCA PSM properties* are intended to support instantiation of SCA applications, and may be used for other purposes.

5.3.1 Structure properties

The *structure properties*, specified in section 4.2 of *Transceiver Facility PIM Specification* [Ref1], specify aspects related to the structure of a *transceiver* implementation.

Structure properties map to *SCA capability properties*, with same consistency conditions as in the *PIM specification*.

The PIM simple *structure properties* **map to** the following *SCA capability properties*:

PIM *structure property*	*SCA capability property*	SCA type (Reserved values)
TX_CHANNELS	txChannels	*unsigned short*
RX_CHANNELS	rxChannels	*unsigned short*
DUPLEX	duplex	*long* 1: fullDuplex 2: halfDuplex
TX_SHAPING	txShaping	*long* 1: nominal 2: specific
TIME_COUPLING	timeCoupling	*long* 1: autonomous 2: coupled 3: coupledToTerminalTime

Table 8 ***SCA capability properties*** **for simple** ***structure properties***

The PIM *structure property* **TX_SERVICES maps to** the following *SCA capability properties*:

SCA capability property	SCA type
hasTxServicesReset	*boolean*
hasTxServicesRadioSilence	*boolean*
hasTxServicesDirectCreation	*boolean*
hasTxServicesRelativeCreation	*boolean*
hasTxServicesAbsoluteCreation	*boolean*
hasTxServicesStrobedCreation	*boolean*
hasTxServicesTermination	*boolean*
hasTxServicesInitialTuning	*boolean*
hasTxServicesInitialRetuning	*boolean*
hasTxServicesGainLocking	*boolean*
hasTxServicesTimeAccess	*boolean*
hasTxServicesApplicationStrobe	*boolean*
hasTxServicesEvents	*boolean*
hasTxServicesErrors	*boolean*
hasTxGainChanges	*boolean*

Table 9 ***SCA capability properties*** **for TX_SERVICES** ***structure property***

The PIM *structure property* **RX_SERVICES maps to** the following *SCA capability properties*:

SCA *capability property*	SCA type
hasRxServicesReset	boolean
hasRxServicesRadioSilence	boolean
hasRxServicesDirectCreation	boolean
hasRxServicesRelativeCreation	boolean
hasRxServicesAbsoluteCreation	boolean
hasRxServicesStrobedCreation	boolean
hasRxServicesTermination	boolean
hasRxServicesInitialTuning	boolean
hasRxServicesInitialRetuning	boolean
hasRxServicesGainLocking	boolean
hasRxServicesTimeAccess	boolean
hasRxServicesApplicationStrobe	boolean
hasRxServicesEvents	boolean
hasRxServicesErrors	boolean
hasRxGainChanges	boolean

Table 10 ***SCA capability properties* for RX_SERVICES *structure property***

5.3.2 Behavior properties

The *behavior properties*, specified in section 4.3 of *Transceiver Facility PIM Specification* [Ref1], specify aspects related to the behavior of a *transceiver* instance.

For *behavior properties* whose nature is specified as *SCA configure properties*, a *transceiver* implementation may only support a limited set out of the specified reserved values. A call to *configure()* with a non-supported reserved value generates an error.

The *behavior properties* **map to** the following *SCA PSM properties*:

PIM *behavior property*	SCA property nature	*SCA PSM property*	SCA type (Reserved values)
TUNING_ASSOCIATION	Configure	tuningAssociation	long 1: sequential 2: burstReferencing
AGC	Configure	Agc	long 1: noAGC 2: earlyControl 3: permanentControl
ALC	Configure	Alc	long 1: noALC 2: activeALC
TUNING_TIMEOUT	Capability	tuningTimeout	unsigned long
1ST_SAMPLE_TIMEOUT	Capability	1stSampleTimeout	unsigned long

Table 11 ***SCA PSM properties* for *behavior properties***

5.3.3 Notification properties

The *notification properties*, specified in section 4.4 of *Transceiver Facility PIM Specification* [Ref1], specify aspects relative to notifications made by a *transceiver* instance to the *radio application*.

The simple *notification properties* **map to** the following *SCA capability properties*:

PIM *notification property*	*SCA capability property*	SCA type
EXCEPTIONS_SUPPORT	exceptionsSupport	Boolean

Table 12 ***SCA capability properties*** **for simple** ***notification properties***

The PIM *notification property* **EVENTS** **maps to** the following *SCA capability properties*:

SCA capability property	SCA type
eventProcessingStart	Boolean
eventProcessingStop	Boolean
eventSilenceStart	Boolean
eventSilenceStop	Boolean

Table 13 ***SCA capability properties*** **for EVENTS** ***notification properties***

The mapping of the other *notification properties* (**EXCEPTIONS** and **ERRORS**) is *unspecified*.

5.3.4 Interface declaration properties

The *interface declaration properties*, specified in section 4.5 of *Transceiver Facility PIM Specification* [Ref1], specify aspects relative to declarations of *service interfaces*.

The *interface declaration properties* **map to** specific *types* declaration constructs.

In addition, *interface declaration properties* **map to** the following *SCA capability properties*:

PIM *interface declaration property*	*SCA capability property*	SCA type (Reserved values)
TX_META_DATA	txMetaData	Boolean
RX_META_DATA	rxMetaData	Boolean

Table 14 ***SCA capability properties*** **for** ***interface declaration properties***

5.3.5 Initialization properties

The *initialization properties*, specified in section 4.6 of *Transceiver Facility PIM Specification* [Ref1], specify the conditions to be met by a *transceiver* implementation when the **CONFIGURED** state is reached by its *Tx channels* and *Rx channels*.

The *initialization properties* **map to** the following *SCA configure properties*:

PIM *initialization property*	*SCA configure property*	SCA type
`INIT_TX_PACKETS_LENGTH`	`initTxPacketsLength`	`unsigned long`
`INIT_CARRIER_FREQ`	`initCarrierFreq`	`unsigned long long`
`INIT_GAIN`	`initGain`	`short`

Table 15 ***SCA configure properties*** **for** ***initialization properties***

5.3.6 Storage properties

The *storage properties*, specified in section 4.9 of *Transceiver Facility PIM Specification* [Ref1], specify the number of calls to certain operations a *transceiver* instance can store before blocking further calls until storage becomes available again.

A call to *configure()* with a requested value outside the range of what the implementation supports generates an error.

The *storage properties* **map to** the following *SCA configure properties*:

PIM *storage property*	*SCA configure property*	SCA type
`CREATION_STORAGE`	`creationStorage`	`unsigned short`
`TUNING_STORAGE`	`tuningStorage`	`unsigned short`
`TX_BASEBAND_STORAGE`	`txBasebandStorage`	`unsigned long`

Table 16 ***SCA configure properties*** **for** ***storage properties***

5.3.7 Parameters validity properties

The *parameter validity properties*, specified in section 4.7 of *Transceiver Facility PIM Specification* [Ref1], specify the validity conditions applicable to parameters in *service interfaces primitives*.

The *parameter validity properties* **map to** the following *SCA capacity properties* as follows:

PIM *parameter validity property*	*SCA capability properties*	SCA type
MIN_BLOCK_LENGTH	minBlockLength	unsigned long
MAX_BLOCK_LENGTH	maxBlockLength	unsigned long
ALETERNATE_REFERENCING	alternateReferencing	Boolean
MIN_FROM_PREVIOUS	minFromPrevious	unsigned long
MAX_FROM_PREVIOUS	maxFromPrevious	unsigned long
MIN_FROM_STROBE	minFromStrobe	unsigned long
MAX_FROM_STROBE	maxFromStrobe	unsigned long
MAX_PACKETS_LENGTH	maxPacketsLength	unsigned long
MAX_TUNING_PRESET	maxTuningPreset	unsigned short
MIN_CARRIER_FREQ	minCarrierFreq	unsigned long long
MAX_CARRIER_FREQ	maxCarrierFreq	unsigned long long
MIN_GAIN	minGain	Short
MAX_GAIN	maxGain	Short
MIN_FROM_ONGOING	minFromOngoing	unsigned long
MAX_FROM_ONGOING	maxFromOngoing	unsigned long

Table 17 ***SCA capability properties*** **for** ***parameter validity properties***

5.3.8 WCET properties

The PIM *WCET properties*, specified in section 4.15 of *Transceiver Facility PIM Specification* [Ref1], specify the maximum length of time, in nanoseconds (ns), possibly taken between the invocation and the return of a primitive.

For PIM *WCET properties*, *SCA capability properties* named `<primitiveName>Wcet` can be implemented, of type `unsigned long` (values in ns).

5.3.9 Unspecified mapping

The mapping of the following properties of *Transceiver Facility PIM Specification* [Ref1] is *unspecified*:

- Rapidity properties (section 4.8),
- Levels properties (section 4.10),
- Channelization properties (section 4.11),
- Temporal accuracy properties (section 4.12),
- Invocation lead time properties (section 4.13),
- Invocation delay (section 4.14).

6 References

6.1 Referenced documents

[Ref1] *Transceiver Facility PIM Specification*, The Wireless Innovation Forum, WINNF-TS-0008 V2.1.1, 20 January 2022
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Specifications/WINNF-TS-0008-V2.1.1.pdf

[Ref2] *Principles for WInnForum Facility Standards*, The Wireless Innovation Forum, WINNF-TR-2007, V1.0.0, 13 October 2020
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Reports/WINNF-TR-2007-V1.0.0.pdf

[Ref3] *Software Communications Architecture*
https://www.jtnc.mil/Resources-Catalog/Category/16990/sca/

[Ref4] *WInnForum Facilities PSMs Mapping Rules*, The Wireless Innovation Forum, WINNF-TR-2008, V1.0.1, 18 January 2022
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Reports/WINNF-TR-2008-V1.0.1.pdf

[Ref5] *The Software Communications Architecture Specification v2.2.2*, Joint Program Executive Office (JPEO) - Joint Tactical Radio System (JTRS), 15 May 2006
https://www.jtnc.mil/Resources-Catalog/Category/16990/sca/

[Ref6] *The Software Communications Architecture Specification, version 4.1*, Joint Tactical Networking Center (JTNC), 20 August 2015
https://www.jtnc.mil/Resources-Catalog/Category/16990/sca/

The URLs above were successfully accessed at release date.

END OF THE DOCUMENT