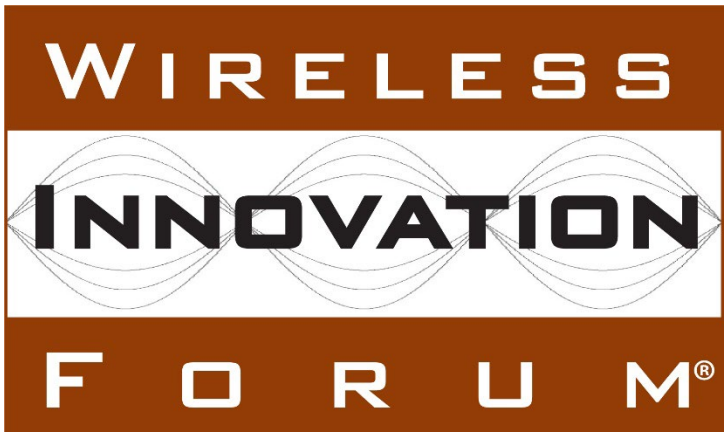

Time Service Facility Native C++ PSM Specification

Document WINNF-TS-3004-App01

Version V1.1.1

18 January 2022

Copyright © 2020-2022 The Software Defined Radio Forum Inc – All Rights Reserved

TERMS, CONDITIONS & NOTICES

This document has been prepared by the Software Defined Systems (SDS) Harmonized Timing Service Task Group 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 It may be amended or withdrawn at a later time and it is not binding on any member of the Forum or of the Harmonized Timing Service Task Group.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.

Copyright © 2020-2022 The Software Defined Radio Forum Inc
All Rights Reserved

Table of Contents

Copyright © 2020-2022 The Software Defined Radio Forum Inc
All Rights Reserved

List of Figures

List of Tables

Copyright © 2020-2022 The Software Defined Radio Forum Inc
All Rights Reserved

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 WInnForum member organizations:

- Marc Adrat, Fraunhofer FKIE,
- Guillaume Delbarre, DGA,
- David Hagood, Cynosure,
- Olivier Kirsch, KEREVAL,
- Francois Levesque, NordiaSoft,
- Charles Linn, L3Harris,
- David Murotake, HiKE,
- Eric Nicollet, Thales,
- Kevin Richardson, MITRE,
- Sarvpreet Singh, Viavi Solutions.

Copyright © 2020-2022 The Software Defined Radio Forum Inc
All Rights Reserved

Time Service Facility Native C++ PSM Specification

1 Introduction

This document WINNF-TS-3004-App01 is the *native C++ PSM specification* of the *WInnForum Time Service Facility* V1.1.0.

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

It addresses the Native C++ programming paradigm, applying the mapping rules of the Native C++ section of *WInnForum Facilities PSMs Mapping Rules* [Ref3] and specifically reporting any deviation to those rules.

The following figure positions the interfaces addressed by the *native C++ PSM specification*:

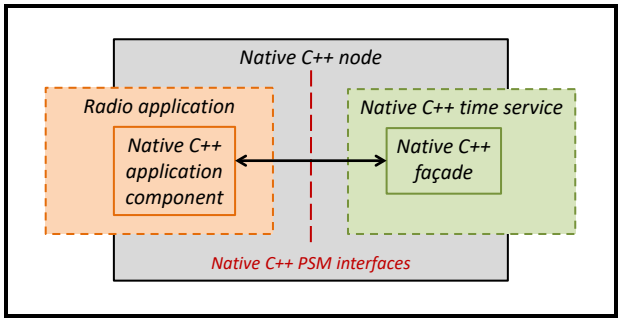

Figure 1 Interfaces addressed by *Time Service Facility native C++ PSM specification*

As depicted, the *native C++ PSM specification* addresses the *native C++ PSM interfaces* of *time services*, positioned, within a *native C++ node*, between the *native C++ application components* of *radio applications* and *native C++ façades* of *native C++ time services*.

The *native C++ PSM specification* addresses two versions of the C++ language:

- C++03, specified by [Ref4], typically applicable for DSP (Digital Signal Processors) processing nodes where compilers do not support recent C++ language versions,
- C++11, specified by [Ref5], suited to state-of-the art programming environments.

It uses the abbreviation "tsf" to identify *Time Service Facility* in formal identifiers.

1.1 Reference definitions

The *Time Service Facility native C++ PSM specification* applies the following definitions from *Time Service Facility PIM Specification* [Ref1]:

Topic	Used definitions
Time service concepts	*time service*, *Time Service Facility*

Table 1 Definitions from *Time Service Facility PIM Specification*

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

The *Time Service Facility native C++ PSM specification* applies the following definitions from *Principles for WInnForum Facility Standards* [Ref2]:

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

Table 2 Definitions from *Principles for WInnForum Facility Standards*

The *Time Service Facility native C++ PSM specification* applies the following definitions from the Native C++ section of *WInnForum Facilities PSMs Mapping Rules* [Ref3]:

Topic	Used definitions
Specification purpose	*native C++ PSM specification*, *native C++ PSM interface*
Software architecture	*native C++ application component, native C++ node*, *native C++ façade*

Table 3 Definitions from Native C++ 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

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

A *native C++ time service* **is defined as** a *time service* implementation with all of its *native C++ façades* being conformant with the *native C++ PSM specification*.

1.2.2 Radio application items

A *native C++ application component* **is conformant with** the *Time Service Facility 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 *time service*.

1.3 Document structure

Section 2 specifies the normative classes for the *native C++ PSM interfaces*.

Section 3 specifies the header files to be used by *native C++ time services*.

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

2 Native C++ PSM classes

This normative section specifies the C++ classes for *native C++ PSM interfaces*.

2.1 Provide and Use Services Interfaces

The *service interfaces* for the *provide services* of the *native C++ PSM interfaces* **are specified by** the following table:

PIM *service interface* (in `TimeService::`)	PIM section	Native C++ PSM *service interface* (in `WInnF_Cpp::TimeService::`)
`TerminalTime::TerminalTimeAccess`	3.1.2	`TerminalTime::TerminalTimeAccess`
`SystemTime::SystemTimeAccess`	3.1.3	`SystemTime::SystemTimeAccess`
`SystemTime::StandardTimeProvision`	3.1.4	`SystemTime::StandardTimeProvision`
`SpecificTimes::SpecificTimeHandling`	3.1.6	`SpecificTimes::SpecificTimeHandling`

Table 4 Provide services mapping

The *service interfaces* for the *use services* of the *native C++ PSM interfaces* **are specified by** the following table:

PIM *service interface* (in `TimeService::`)	PIM section	Native C++ PSM *service interface* (in `WInnF_Cpp::TimeService::`)
`StandardTimes::ReferencesNotification`	3.1.5	`StandardTimes::ReferencesNotification`
`SpecificTimes::SettingsNotification`	3.1.7	`SpecificTimes::SettingsNotification`

Table 5 Use services mapping

2.2 Facade class

The `WInnF_Cpp::TimeService::Facade` class **is specified as** the class derived from the `Facade` class of the Native C++ section of *WInnForum Facilities PSMs Mapping Rules* [Ref3].

Since a *time service* does not feature a `CONFIGURED` state, the `Facade` class does not feature the methods *activeServicesInitialized()* and *activeServicesReleased()*.

Two classes are available for *active services access*, as specified by [Ref3]:

- `ExplicitServicesAccess`,
- `GenericServicesAccess`.

Usage of those classes by *façades* is controlled by the preprocessing variables EXPLICIT_SERVICES_ACCESS and GENERIC_SERVICES_ACCESS.

A *façade* may implement the two possibilities, in which case the two preprocessing variables are set.

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

Typical possibilities for the **`Facade`** class are represented by the following class diagram:

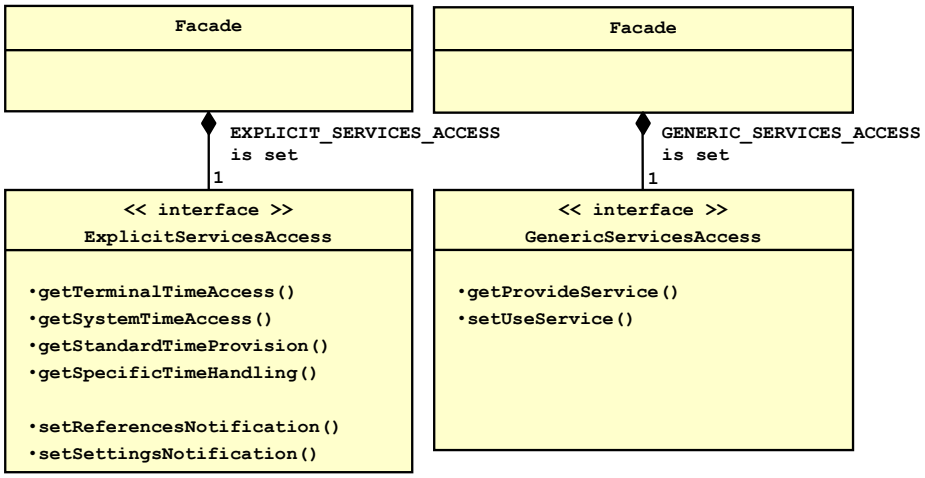

Figure 2 Class diagram of Native C++ Facade

The lifecycle of a **`Facade`** instance is greater than the lifecycle of the *application component* using the *time service*. Other aspects are *unspecified*.

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

3 Native C++ PSM header files

This normative section specifies the standard C++ header files (`.hpp`) to be used by conformant (see section 1.2) *native C++ façades* and *native C++ application components*.

Conformant *native C++ façades* and *native C++ application components* have only to include `TsfFacade.hpp`.

`TsfPlatformProviderAdaptations.hpp` captures the adaptations decided by *native C++ time services* implementers.

The supported C++ versions are C++03 and C++11. The standard macro `__cplusplus` indicates the supported C++ version.

The specified header files have been successfully compiled using:

- GCC 4.8.5 and 7.5.0 for C++ 98, C++ 11, C++14 and C++17.

The used code formatting rules are specified by the Native C++ section of *WInnForum Facilities PSMs Mapping Rules* [Ref3].

3.1 <cstdint> (stdint.h)

The `<cstdint> (stdint.h)` library is used for C++ standard declarations for basic types.

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

3.2 TsfFacade.hpp

The **TsfFacade.hpp** file **is specified as** the header file that declares the **Facade** class.

The content of **TsfFacade.hpp** **is specified as**:

```
#ifndef TSF_FACADE
#define TSF_FACADE

#include "TsfPlatformProviderAdaptations.hpp"
#include "TsfServices.hpp"

#ifdef EXPLICIT_SERVICES_ACCESS
   #include "TsfExplicitServicesAccess.hpp"
#endif

#ifdef GENERIC_SERVICES_ACCESS
   #include "TsfGenericServicesAccess.hpp"
#endif

namespace WInnF_Cpp
{
   namespace TimeService
   {

      // Facade (section 2.2 of [PSM])
      class Facade
      {
       public:
#ifdef EXPLICIT_SERVICES_ACCESS
         virtual ActiveServicesAccess::ExplicitServicesAccess *
         getExplicitServicesAccess() NOEXCEPT = 0;
#endif

#ifdef GENERIC_SERVICES_ACCESS
         virtual ActiveServicesAccess::GenericServicesAccess *
         getGenericServicesAccess() NOEXCEPT = 0;
#endif
         virtual ~Facade() NOEXCEPT {}
      };
   } // namespace TimeService
} // namespace WInnF_Cpp

#endif // ifndef TSF_FACADE
```

3.3 TsfPlatformProviderAdaptations.hpp

The **TsfPlatformProviderAdaptations.hpp** file **is specified as** the header file specifying the **ActiveServicesAccess** property, specified by section 2.2.

TsfPlatformProviderAdaptations.hpp has to be adapted by platform providers only in areas marked with *** PROVIDER-ADAPTATION ***.

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

The content of **`TsfPlatformProviderAdaptations.hpp` is specified as**:

```
// Specifies PlatformProvider-adaptations:
//  ActiveServicesAccess property, specified by section 2.3 of [PSM]

#ifndef TSF_PLATFORM_PROVIDER_ADAPTATIONS
#define TSF_PLATFORM_PROVIDER_ADAPTATIONS

// ActiveServicesAccess
// *** PROVIDER-ADAPTATION ***
#define EXPLICIT_SERVICES_ACCESS // comment if EXPLICIT is not supported
#define GENERIC_SERVICES_ACCESS  // comment if GENERIC is not supported
#define EXCEPTIONS_SUPPORT       // comment if EXCEPTIONS is not supported

#endif // #ifndef TSF_PLATFORM_PROVIDER_ADAPTATIONS
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

3.4 TsfTypes.hpp

The **`TsfTypes.hpp`** file **is specified as** the header file that declares the *API types* specified by section 3.4 of the *Time Service Facility PIM Specification* [Ref1].

The ***`int`*** type for *SpecificTimeId* in the *PIM specification* (not a valid IDL type) **maps to** ***`unsigned 16-bit`***.

The ***`int`*** type for *SourceId* in the *PIM specification* (not a valid IDL type) **maps to** ***`unsigned 8-bit`***.

The content of **`TsfTypes.hpp`** **is specified as**:

```
#ifndef TSF_TYPES
#define TSF_TYPES

#if __cplusplus >= 201103L // C++2011
  #include <cstdint>
#else
  #include <stdint.h>
#endif

#include "TsfPlatformProviderAdaptations.hpp"
// Support of exceptions (section 3.8 of [PIM])
#ifdef EXCEPTIONS_SUPPORT
   #include "TsfExceptions.hpp"
#else
   // If exceptions are not supported, make sure NOEXCEPT is defined
   #ifndef NOEXCEPT
      #if __cplusplus < 201103L
         #define NOEXCEPT
      #else
         #define NOEXCEPT noexcept
      #endif
   #endif
#endif

namespace WInnF_Cpp
{
   namespace TimeService
   {

      // SourceId (signed int in the [PIM], no dedicated type)
      typedef uint8_t SourceId;

      // SpecificTimeId (signed int in the [PIM], no dedicated type)
      typedef uint16_t SpecificTimeId;

      // TimeValue (section 3.4.1 of [PIM])
      struct TimeValue
      {
         uint32_t seconds;     // in seconds
         uint32_t nanoseconds; // in nanoseconds (<1.000.000.000)
      };

      const TimeValue UndefinedTime = {0xFFFFFFFF, 0xFFFFFFFF};

      // TimeUncertainty (section 3.4.2 of [PIM])
      typedef int32_t TimeUncertainty;
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

```
      const TimeUncertainty Beyond2SecTimeUncertainty     = 0xFFFFFFF0;
      const TimeUncertainty Beyond4SecTimeUncertainty     = 0xFFFFFFF1;
      const TimeUncertainty Beyond8SecTimeUncertainty     = 0xFFFFFFF2;
      const TimeUncertainty Beyond16SecTimeUncertainty    = 0xFFFFFFF3;
      const TimeUncertainty Beyond32SecTimeUncertainty    = 0xFFFFFFF4;
      const TimeUncertainty Beyond64SecTimeUncertainty    = 0xFFFFFFF5;
      const TimeUncertainty Beyond128SecTimeUncertainty   = 0xFFFFFFF6;
      const TimeUncertainty Beyond256SecTimeUncertainty   = 0xFFFFFFF7;
      const TimeUncertainty Beyond512SecTimeUncertainty   = 0xFFFFFFF8;
      const TimeUncertainty Beyond1024SecTimeUncertainty  = 0xFFFFFFF9;
      const TimeUncertainty Beyond2048SecTimeUncertainty  = 0xFFFFFFFA;
      const TimeUncertainty Beyond4096SecTimeUncertainty  = 0xFFFFFFFB;
      const TimeUncertainty Beyond8192SecTimeUncertainty  = 0xFFFFFFFC;
      const TimeUncertainty Beyond16384SecTimeUncertainty = 0xFFFFFFFD;

      const TimeUncertainty UnknownTimeUncertainty   = 0xFFFFFFFE;
      const TimeUncertainty UndefinedTimeUncertainty = 0xFFFFFFFF;

      // RateUncertainty (section 3.4.3 of [PIM])
      typedef int32_t RateUncertainty;

      const RateUncertainty UnknownRateUncertainty = 0xFFFFFFFF;
   } // namespace TimeService
} // namespace WInnF_Cpp
#endif // #ifndef TSF_TYPES
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

3.5 TsfServices.hpp

The **`TsfServices.hpp`** file **is specified as** the header file that declares the *service interfaces* specified by section 3.1 of the *Time Service Facility PIM Specification* [Ref1].

It declares **`TSF_PIM_VERSION`** in accordance with section "Referenced PIM version" of the Native C++ section of *WInnForum Facilities PSMs Mapping Rules* [Ref3].

The content of **`TsfServices.hpp`** **is specified as**:

```
#ifndef TSF_SERVICES
#define TSF_SERVICES

#if __cplusplus < 199711L
   #error C++ version must be C++98 or later
#endif

// TSF_PIM_VERSION property
#define TSF_PIM_VERSION 0x010100L // For V1.1.0

#include "TsfTypes.hpp"

namespace WInnF_Cpp
{
   namespace TimeService
   {
      namespace TerminalTime
      {
         // TerminalTime::TerminalTimeAccess (section 3.1.2 of [PIM])
         class TerminalTimeAccess
         {
          public:
            virtual void getTerminalTime(
                TimeValue *terminalTime) const NOEXCEPT = 0;
            virtual void getTerminalTimeRateUncertainty(
                RateUncertainty *terminalTimeRateUncertainty)
                const NOEXCEPT = 0;
            virtual ~TerminalTimeAccess() NOEXCEPT {}
         };
      } // namespace TerminalTime

      namespace SystemTime
      {
         // SystemTime::SystemTimeAccess (section 3.1.3 of [PIM])
         class SystemTimeAccess
         {
          public:
            virtual void getCurrentTAI(
                TimeValue *      currentTAI,
                TimeValue *      timeStamp,
                TimeUncertainty *timeUncertainty) const NOEXCEPT = 0;
            virtual void getCurrentUTC(
                TimeValue *      currentUTC,
                TimeValue *      timeStamp,
                TimeUncertainty *timeUncertainty) const NOEXCEPT = 0;
            virtual void getLastUpdateTAI(
                TimeValue *      lastUpdateTAI,
                TimeValue *      timeStamp,
                TimeUncertainty *timeUncertainty) const NOEXCEPT = 0;
            virtual void getLastUpdateUTC(
                TimeValue *      lastUpdateUTC,
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

```
            TimeValue *      timeStamp,
            TimeUncertainty *timeUncertainty) const NOEXCEPT = 0;
         virtual ~SystemTimeAccess() NOEXCEPT {}
      };

      // SystemTime::StandardTimeProvision (section 3.1.4 of [PIM])
      class StandardTimeProvision
      {
       public:
         virtual void provideTAI(TimeValue       providedTAI,
                                 TimeValue       timeStamp,
                                 TimeUncertainty timeUncertainty,
                                 SourceId        sourceId) = 0;
         virtual void provideUTC(TimeValue       providedUTC,
                                 TimeValue       timeStamp,
                                 TimeUncertainty timeUncertainty,
                                 SourceId        sourceId) = 0;
         virtual ~StandardTimeProvision() NOEXCEPT {}
      };
   } // namespace SystemTime

   namespace StandardTimes
   {
      // StandardTimes::ReferencesNotification (section 3.1.5 of [PIM])
      class ReferencesNotification
      {
       public:
         virtual void notifyStandardTimeReference(
            TimeValue       referenceTAI,
            TimeValue       referenceUTC,
            TimeValue       timeStamp,
            TimeUncertainty timeUncertainty,
            SourceId        sourceId) NOEXCEPT = 0;
         virtual ~ReferencesNotification() NOEXCEPT {}
      };
   } // namespace StandardTimes

   namespace SpecificTimes
   {
      // SpecificTimes::SpecificTimeHandling (section 3.1.6 of [PIM])
      class SpecificTimeHandling
      {
       public:
         virtual void setSpecificTime(SpecificTimeId  specificTimeId,
                                      TimeValue       specificTime,
                                      TimeValue       timeStamp,
                                      TimeUncertainty timeUncertainty) = 0;
         virtual void getSpecificTime(
            SpecificTimeId   specificTimeId,
            TimeValue *      specificTime,
            TimeValue *      timeStamp,
            TimeUncertainty *timeUncertainty) const = 0;
         virtual ~SpecificTimeHandling() NOEXCEPT {}
      };


      // SpecificTimes::SettingsNotification (section 3.1.7 of [PIM])
      class SettingsNotification
      {
       public:
         virtual void notifySpecificTimeSetting(
            SpecificTimeId  specificTimeId,
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

```
                TimeValue       specificTime,
                TimeValue       timeStamp,
                TimeUncertainty timeUncertainty) NOEXCEPT = 0;
            virtual ~SettingsNotification() NOEXCEPT {}
         };
      } // namespace SpecificTimes
   }    // namespace TimeService
} // namespace WInnF_Cpp

#endif // ifndef TSF_SERVICES
```

3.6 TsfExplicitServicesAccess.hpp

The content of **`TsfExplicitServicesAccess.hpp`** **is specified as**:

```
#ifndef TSF_EXPLICIT_SERVICES_ACCESS
#define TSF_EXPLICIT_SERVICES_ACCESS

namespace WInnF_Cpp
{
   namespace TimeService
   {
      namespace ActiveServicesAccess
      {
         // Access to specific active services
         class ExplicitServicesAccess
         {
          public:
            // Provide services instances
            // TerminalTime services group
            virtual TerminalTime::TerminalTimeAccess *getTerminalTimeAccess()
                = 0;

            // SystemTime services group
            virtual SystemTime::SystemTimeAccess *getSystemTimeAccess() = 0;
            virtual SystemTime::StandardTimeProvision *
            getStandardTimeProvision() = 0;

            // SpecificTimes services group
            virtual SpecificTimes::SpecificTimeHandling *
            getSpecificTimeHandling() = 0;

            // Use services
            // StandardTimes services group
            virtual void setReferencesNotification(
                StandardTimes::ReferencesNotification *reference) = 0;

            // SpecificTimes services group
            virtual void setSettingsNotification(
                SpecificTimes::SettingsNotification *reference) = 0;

            virtual ~ExplicitServicesAccess() NOEXCEPT {}
         };
      } // namespace ActiveServicesAccess
   } // namespace TimeService
} // namespace WInnF_Cpp
#endif // ifndef TSF_EXPLICIT_SERVICES_ACCESS
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

3.7 TsfGenericServicesAccess.hpp

The content of **`TsfGenericServicesAccess.hpp`** **is specified as**:

```
#ifndef TSF_GENERIC_SERVICES_ACCESS
#define TSF_GENERIC_SERVICES_ACCESS

#include "TsfTypes.hpp"

namespace WInnF_Cpp
{
   namespace TimeService
   {
      namespace ActiveServicesAccess
      {
         class Object
         {
          public:
            virtual ~Object() NOEXCEPT {}
         };

         class GenericServicesAccess
         {
          public:
            // Access to active provide services
            virtual Object *getProvideService(
                const char *serviceName) = 0;

            // Access to active use services
            virtual void setUseService(const char *serviceName,
                                       Object *    useService) = 0;

            virtual ~GenericServicesAccess() NOEXCEPT {}
         };
      } // namespace ActiveServicesAccess
   }    // namespace TimeService
} // namespace WInnF_Cpp
#endif // ifndef TSF_GENERIC_SERVICES_ACCESS
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

3.8 TsfExceptions.hpp

The **TsfExceptions.hpp** file **is specified as** the header file that declares all useable *exceptions*.

TsfExceptions.hpp is used if **EXCEPTION_USE** is equal to **true.**

The content of **TsfExceptions.hpp** **is specified as**:

```
#ifndef TSF_EXCEPTIONS
#define TSF_EXCEPTIONS

#include <stdexcept>

#ifndef NOEXCEPT
   #if __cplusplus < 201103L
      #define NOEXCEPT
   #else
      #define NOEXCEPT noexcept
   #endif
#endif

// Exceptions identifiers (section 3.2 of [PIM])
namespace WInnF_Cpp
{
   namespace TimeService
   {
      // General exceptions
      class FutureTimeStampException : public std::invalid_argument
      {
       public:
         FutureTimeStampException(char const *msg = "") NOEXCEPT
             : ::std::invalid_argument(msg)
         {
         }
         FutureTimeStampException(::std::string const &msg) NOEXCEPT
             : ::std::invalid_argument(msg)
         {
         }
      };

      // Range exceptions
      class InvalidSpecificTimeId : public std::range_error
      {
       public:
         InvalidSpecificTimeId(char const *msg = "") NOEXCEPT
             : ::std::range_error(msg)
         {
         }
         InvalidSpecificTimeId(::std::string const &msg) NOEXCEPT
             : ::std::range_error(msg)
         {
         }
      };

      // Services access exceptions - specified by Native C++ Mapping Rules
      class UnavailableServiceException : public std::runtime_error
      {
       public:
         UnavailableServiceException(char const *msg = "") NOEXCEPT
             : ::std::runtime_error(msg)
         {
         }
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

```
        UnavailableServiceException(::std::string const &msg) NOEXCEPT
            : ::std::runtime_error(msg)
        {
        }
    };
    class InvalidReferenceException : public std::runtime_error
    {
     public:
        InvalidReferenceException(char const *msg = "") NOEXCEPT
            : ::std::runtime_error(msg)
        {
        }
        InvalidReferenceException(::std::string const &msg) NOEXCEPT
            : ::std::runtime_error(msg)
        {
        }
    };
  } // namespace TimeService
} // namespace WInnF_Cpp

#endif // ifndef TSF_EXCEPTIONS
```

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

4 References

4.1 Referenced documents

[Ref1] *Time Service Facility PIM Specification*, The Wireless Innovation Forum, WINNF-TS-3004, V1.1.1, 18 January 2022
https://sds.wirelessinnovation.org/specifications-and-recommendations
https://winnf.memberclicks.net/assets/work_products/Specifications/WINNF-TS-3004-V1.1.1.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] *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

[Ref4] *Information technology – Programming languages – C++*, ISO/IEC 14882:2003
http://www.cplusplus.com/

[Ref5] *Information technology – Programming languages – C++*, ISO/IEC 14882:2011
http://www.cplusplus.com/

The URLs above were successfully accessed at release date.

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved

END OF THE DOCUMENT

Copyright © 2020-2022 The Software Defined Radio Forum Inc.
All Rights Reserved