Joint Tactical Radio System Standard
Timing Service
Application Program Interface

Version: 1.4.4
26 June 2013

Statement A- Approved for public release; distribution is unlimited (17 July 2013).

Revision History

Version	Description	Last Modified Date
0.1	Initial draft release	30-Jun-2006
0.2	Updated draft release	11-Sept-2006
1.0	Updated based on NESI comments	10-Oct-2006
1.1	Revised all interfaces based on comments received during JPEO JTRS Standards conference (Oct 2006)	06-Nov-2006
1.2	Updated based on Working Panel discussions and NESI comments	06-Dec-2006
1.3	Added Appendix A.E – Terminal Time Timestamp Latency Specification, extended TFOM valid ranges in primitive descriptions to include full range, changed all IDL to camel back notation. **ICWG Approved**	13-Dec-2006
1.3.1	Added missing comma to getWfTimeData operation in TimingServiceWfTimeExt.idl	02-May-2007
1.4 <Draft>	Errata: -Corrected the #ifndef/#define in “TimingService.idl” Change Proposals: -Added Section D, System Time Extension.	20-Aug-2007
1.4	Updates based on ICWG review	01-Oct-2007
1.4.1	Updates based on ICWG: - Add missing base port names to all extension port diagrams. **ICWG Approved**	11-Oct-2007
1.4.2	Updated document stamp for public release	30-May-2008
1.4.3 <Draft>	-Update A.5.4.1TimeSpec descriptions -Update B.5.4.1 time descriptions	24-Oct-2011
1.4.3 <Final Draft>	Clarify systemTime and terminalTime descriptions in A.3.2.1, A.3.2.2 and D.3.1.1	10-Jan-2012
1.4.3	Misc. Redline Changes **ICWG Approved**	24-Jan-2012
1.4.4	Preparation for public release **ICWG Approved**	26-Jun-2013

Table of Contents

Table of Contents

Lists of Figures

List of Tables

A. TIMING SERVICE

A.1 INTRODUCTION

The *Timing Service* supports methods and attributes that are specific to the Joint Tactical Radio (JTR) set time it represents.

The *Timing Service* component(s) maintains, manages, and distributes time within the JTR set. This includes Terminal Time and System Time management and distribution. The *Timing Service* provides an interface for retrieving System Time and the quality indicator, i.e. Time Figure of Merit (TFOM), for Terminal and System Time.

Terminal Time is the time returned from the Portable Operating System Interface (POSIX) for time [1] and is monotonic increasing. Terminal Time is used for communicating time among the different terminal components (including distributed processor software and hardware components). The *Timing Service* synchronizes the Terminal Time between distributed components within the terminal. The *Timing Service* controls the local processor's POSIX clock.

System Time is the terminal's estimate of Coordinated Universal Time (UTC) time. UTC time can be derived from various combinations of inputs (e.g. the Global Positioning System (GPS) device, the chronometer device, or operator input) while utilizing the local timing pulse.

This API provides information to the software developer to utilize the *Timing Service* interfaces in the waveform target configurations.

The *Timing Service* defines a SCA provides port [2], *TimingService::TerminalTimeAccess* interface, as the mechanism by which waveform may retrieve Terminal Time TFOM. A waveform retrieves Terminal Time via the POSIX time interfaces [1].

The *Timing Service* also defines a SCA provides port, *TimingService::SystemTimeAccess* interface, as the mechanism by which a waveform may retrieve System Time data.

It is assumed that each waveform component that interfaces with the *Timing Service* with the intention of sending data to the *Timing Service* will implement a SCA uses port [2].

A.1.1 Overview

This base contains as follows:

a. Section A.1, *Introduction*, of this document contains the introductory material regarding the overview, Service Layer Description, Modes, States and Referenced Documents of this document.

b. Section A.2, *Services*, provides summary of service interface uses, interface for each service component, port connections, and sequence diagrams.

c. Section A.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Timing Service*.

d. Section A.4 *IDL*.

e. Section A.5, UML, specifies the data types and exceptions that are provided by the *Timing Service*.

f. Appendix A.A – *Abbreviations and Acronyms.*

g. Appendix A.B – *Performance Specification.*

h. Appendix A.C –*Terminal Time Distribution Example*.

i. Appendix A.D – *TFOM and ETE Values*.

j. Appendix A.E – *Terminal Time Timestamp Latency Specification.*

A.1.2 Service Layer Description

A.1.2.1 Timing Service Port Connections

Figure 1 shows the port connections for the *Timing Service*.

Note: All port names are for reference only.

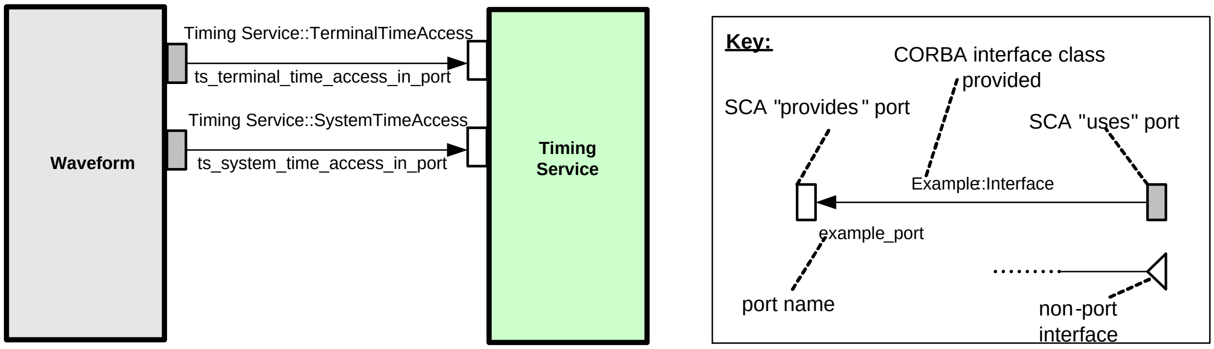
Waveform
Timing Service::TerminalTimeAccess
ts_terminal_time_access_in_port
Timing Service::SystemTimeAccess
ts_system_time_access_in_port
Timing Service
Key:
CORBA interface class provided
SCA "provides" port
SCA "uses" port
Example:Interface
example_port
port name
non-port interface

Figure 1 – Timing Service Port Diagram

***Timing Service* Provides Ports Definitions**

ts_terminal_time_access_in_port permits a waveform to retrieve Terminal Time TFOM.

ts_system_time_access_in_port permits a waveform to retrieve the current System Time, associated Terminal Time timestamp and current System Time TFOM.

***Timing Service* Uses Ports Definitions**

None

A.1.2.2 Modes of Service

None

A.1.3 Referenced Documents

The following documents of the exact issue shown from a part of this specification to the extent specified herein.

A.1.3.1 Government Documents

The following documents are part of this specification as specified herein.

A.1.3.2 Specifications

A.1.3.2.1.1 Federal Specifications

None

A.1.3.2.1.2 Military Specifications

None

A.1.3.3 Other Government Agency Documents

[1] JTRS Standard, "Software Communications Architecture Specification, Appendix B, SCA Application Environment Profile," JPEO, Version 2.2.2.

[2] JTRS Standard, "Software Communications Architecture Specification," JPEO, Version 2.2.2.

A.2 SERVICES

A.2.1 Provide Services

The *Timing Service* provides services consist of the following service ports, interfaces, and primitives, which can be called by other client components.

Table 1 – Timing Service Provide Service Interface

Service Group (Port Name)	Service (Interface Provided)	Primitives (Provided)
ts_terminal_time_access_in_port	TimingService::TerminalTimeAccess	getTerminalTimeTfom()
ts_system_time_access_in_port	TimingService::SystemTimeAccess	getSystemTime()
		getSystemTimeTfom()

A.2.2 Use Services

None

A.2.3 Interface Modules

A.2.3.1 TimingService

The class diagram for the *Timing Service* is shown in Figure 2.

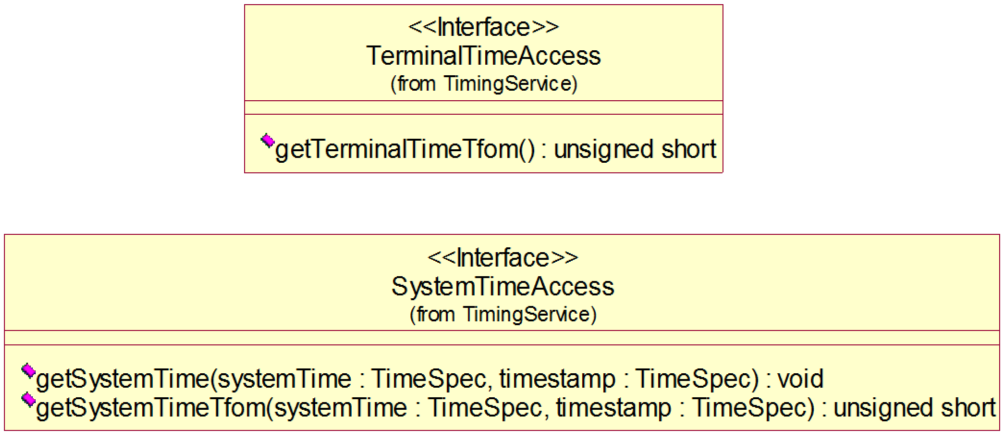
<<Interface>>
TerminalTimeAccess
(from TimingService)
getTerminalTimeTfom() : unsigned short
<<Interface>>
SystemTimeAccess
(from TimingService)
getSystemTime(systemTime : TimeSpec, timestamp : TimeSpec) : void
getSystemTimeTfom(systemTime : TimeSpec, timestamp : TimeSpec) : unsigned short

Figure 2 – Timing Service Class Diagram

A.2.3.2 TerminalTimeAccess Interface Description

The interface design of *TerminalTimeAccess* is shown in Figure 3. It provides the mechanism for waveforms to retrieve Terminal Time TFOM.

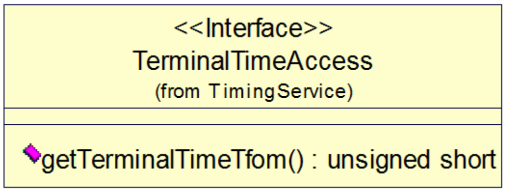

Figure 3 – TerminalTimeAccess Interface Diagram

A.2.3.3 SystemTimeAccess Interface Description

The interface design of *SystemTimeAccess* is shown in Figure 4. It provides the mechanism for waveforms to retrieve System Time data.

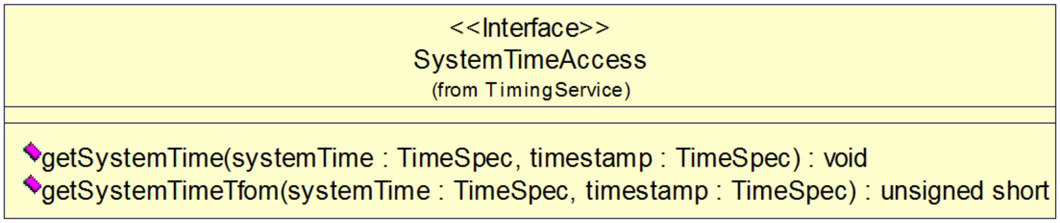
<<Interface>>
SystemTimeAccess
(from TimingService)
getSystemTime(systemTime : TimeSpec, timestamp : TimeSpec) : void
getSystemTimeTfom(systemTime : TimeSpec, timestamp : TimeSpec) : unsigned short

Figure 4 – SystemTimeAccess Interface Diagram

A.2.4 Sequence Diagrams

None

A.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in section A.5 UML. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.

A.3.1 TimingService::TerminalTimeAccess

A.3.1.1 *getTerminalTimeTfom* Operation

This operation provides the ability to retrieve the local processor's Terminal Time TFOM from the *Timing Service* valid at the next local timing pulse.

A.3.1.2 Synopsis

unsigned short getTerminalTimeTfom ( );

A.3.1.3 Parameters

None

A.3.1.4 State

This operation is only valid during the ENABLED state.

A.3.1.5 New State

Not Applicable

A.3.1.6 Return Value

Description	Type	Units	Valid Range
The Terminal Time TFOM	unsigned short	A numerical value indicating the accuracy of the Terminal Time. *See Appendix A.D – TFOM and ETE Values for each TFOM numerical value's associated range of Estimated Time Error (ETE).*	1 – 15

A.3.1.7 Originator

Waveform

A.3.1.8 Exceptions

None

A.3.2 TimingService::SystemTimeAccess

A.3.2.1 *getSystemTime* Operation

This operation provides the ability to retrieve the current System Time and the associated Terminal Time stamp from the *Timing Service*.

A.3.2.2 Synopsis

void getSystemTime (out TimeSpec systemTime, out TimeSpec timestamp);

A.3.2.3 Parameters

Parameter Name	Description	Type	Units	Valid Range
systemTime	The System Time value returned to the calling waveform. Value returned represent the amount of time since epoch 1 Jan 2000 00:00:00 ZULU.	TimeSpec	time	See Section A.5.4.1.
timestamp	The associated Terminal Time corresponding to the time when the System Time was valid. *See Appendix A.E – Terminal Time Timestamp Latency Specification*	TimeSpec	time	See Section A.5.4.1.

A.3.2.4 State

This operation is only valid during the ENABLED state.

A.3.2.5 New State

Not applicable

A.3.2.6 Return Value

None

A.3.2.7 Originator

Waveform

A.3.2.8 Exceptions

None

A.3.2.9 *getSystemTimeTfom* Operation

This operation provides the ability to retrieve the current System Time TFOM, the current System Time, and the associated Terminal Time timestamp from the *Timing Service*.

A.3.2.10 Synopsis

unsigned short getSystemTimeTfom (out TimeSpec systemTime, out TimeSpec timestamp);

A.3.2.11 Parameters

Parameter Name	**Description**	**Type**	**Units**	**Valid Range**
systemTime	The System Time value returned to the calling waveform. Value returned represent the amount of time since epoch 1 Jan 2000 00:00:00 ZULU.	TimeSpec	time	See Section A.5.4.1.
timestamp	The associated Terminal Time corresponding to the time when the System Time was valid. *See Appendix A.E – Terminal Time Timestamp Latency Specification*	TimeSpec	time	See Section A.5.4.1.

A.3.2.12 State

This operation is only valid during the ENABLED state.

A.3.2.13 New State

Not applicable

A.3.2.14 Return Value

Description	**Type**	**Units**	**Valid Range**
The System Time TFOM	unsigned short	A numerical value indicating the accuracy of the System Time. *See Appendix A.D – TFOM and ETE Values for each TFOM numerical value's associated range of ETE.*	1 – 15

A.3.2.15 Originator

Waveform

A.3.2.16 Exceptions

None

A.4 IDL

A.4.1 TimingService

```
/*
** TimingService.idl
*/

#ifndef __TIMINGSERVICE_DEFINED
#define __TIMINGSERVICE_DEFINED

module TimingService
{

    typedef long SecType;
    typedef long NSecType;

    struct TimeSpec
    {
        SecType  sec;
        NSecType nsec;
    };

    interface TerminalTimeAccess
    {

        unsigned short getTerminalTimeTfom ( );

    };

    interface SystemTimeAccess
    {

        void getSystemTime (out TimeSpec systemTime,
                            out TimeSpec timestamp);

        unsigned short getSystemTimeTfom (out TimeSpec systemTime,
                                          out TimeSpec timestamp);

    };

};

#endif // __TIMINGSERVICE_DEFINED
```

A.5 UML

This section contains the service component UML diagram and the definitions of all data types referenced (directly or indirectly) by section A.3 Service Primitives and Attributes.

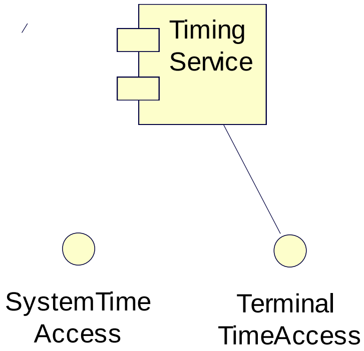

Figure 5 – Timing Service Component Diagram

A.5.1 Data Types

A.5.1.1 TimingService::SecType Typedef

The *SecType* type definition defines a type used to declare a second of time. It is of type long.

```
typedef long SecType;
```

A.5.1.2 TimingService::NSecType Typedef

The *NSecType* type definition defines a type used to declare a nanosecond of time. It is of type long.

```
typedef long NSecType;
```

A.5.2 Enumerations

None

A.5.3 Exceptions

None

A.5.4 Structures

A.5.4.1 TimingService::TimeSpec Structure

The *TimeSpec* structure defines the time specification type for primitives utilizing time.

```
struct TimeSpec
{
        SecType sec;
        NSecType nsec;
};
```

Struct	Attributes	Type	Valid Range	Description
TimeSpec	sec	SecType (See Section 0)	-2147483648 to 2147483647	Signed seconds portion of time.
	nsec	NSecType (See Section A.5.1.2)	0 to 999999999	Signed nanoseconds portion of time.

APPENDIX A.A – ABBREVIATIONS AND ACRONYMS

API	Application Program Interface
CORBA	Common Object Request Broker Architecture
ETE	Estimated Time Error
GPS	Global Positioning System
ICWG	Interface Control Working Group
ID	Identifier
IDL	Interface Definition Language
ISR	Interrupt Service Routine
JPEO	Joint Program Executive Office
JTNC	Joint Tactical Networking Center
JTR	Joint Tactical Radio
JTRS	Joint Tactical Radio System
ms	Millisecond
N/A	Not Applicable
ns	Nanoseconds
NESI	Net-centric Enterprise Solutions for Interoperability
POSIX	Portable Operating System Interface
s	Seconds
SCA	Software Communications Architecture
TFOM	Time Figure of Merit
TTW	Terminal Time Word
UML	Unified Modeling Language
us	Microsecond
UTC	Universal Time Coordinated

APPENDIX A.B – PERFORMANCE SPECIFICATION

Table 2 provides a template for the generic performance specification for the *Timing Service* API which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 1.

Table 2 – Timing Service Performance Specification

Specification	**Description**	**Units**	**Value**
Worst Case Command Execution Time for ts_terminal_time_access_in_port	*	*	*
Worst Case Command Execution Time for ts_system_time_access_in_port	*	*	*

Note: (*) These values should be filled in by individual developers.

APPENDIX A.C –TERMINAL TIME DISTRIBUTION EXAMPLE

The following is for reference only and guidance.

The *Timing Service* component(s) may be centralized in one *Timing Service* or distributed with many *Timing Services*.

Figure 6 shows an example of how Terminal Time may be distributed throughout the JTR set. The *Timing Service* maintains and distributes Terminal Time to all interested processing elements (e.g. via the Event Service). The distributed time is the Terminal Time valid at the next local timing pulse.

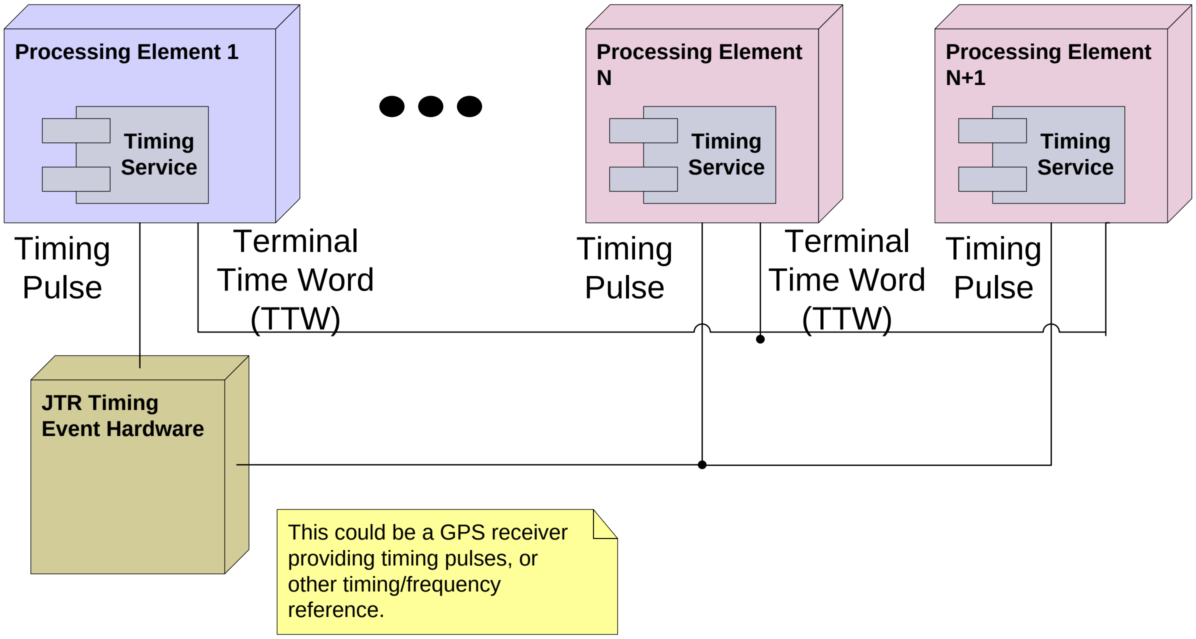

Figure 6 – JTR Terminal Time Distribution Example

Another aspect of the distribution of Terminal Time is the client portion of the *Timing Service*. This client portion will implement the interface via which to receive the Terminal Time from the waveform, and also the interrupt service routine (ISR) which will execute upon detection of the local timing pulse and will be responsible for "jam-setting" the Terminal Time (valid at the timing pulse) into the local operating system via a POSIX system call. This client portion will have to be resident on every CORBA capable processor wishing to utilize the Terminal Time provided by the *Timing Service*.

APPENDIX A.D – TFOM AND ETE VALUES

Table 3 – TFOM Numerical Values and Estimated Errors

TFOM	Estimated Time Error (ETE, UTC)
0	(Note 1)
1	ETE ≤ 1 ns
2	1 ns < ETE ≤ 10 ns
3	10 ns < ETE ≤ 100 ns
4	100 ns < ETE ≤ 1 us
5	1 us < ETE ≤ 10 us
6	10 us < ETE ≤ 100 us
7	100 us < ETE ≤ 1 ms
8	1 ms ≤ ETE ≤ 10 ms
9	10 ms < ETE, or Fault
10-14	Not used
15	ETE Not Available

Note 1: External time source indicates proper/normal operation by TFOM = 0.

APPENDIX A.E – TERMINAL TIME TIMESTAMP LATENCY SPECIFICATION

Table 4 provides a template for the generic latency specification for the associated Terminal Time *timestamp* parameter returned from *Timing Service* API primitives that will be documented in the service or device using the interface.

Table 4 – Terminal Time Timestamp Latency Specification

Specification	**Description**	**Unit**	**Value**
Worst Case Latency for Terminal Time for timestamp	*	*	*

Note: (*) These values should be filled in by individual developers.

B. EXTERNAL TIME REFERENCE (ETR) EXTENSION

B.1 INTRODUCTION

The *External Time Reference (ETR) Extension* is based upon the *Timing Service* API. It extends the functionality of the *Timing Service* to support the collection and distribution of ETR data. It retains the methods and attributes defined in the base section A Timing Service API.

The *ETR Extension* defines a SCA provides port [2], *TimingService::EtrAccess* interface, as the mechanism for waveforms to enable or disable the output of an ETR event (via the *TimingService::EtrEvent* interface) when an ETR event occurs.

The *ETR Extension* also defines a non-SCA uses port, *TimingService::EtrEvent* interface, as the mechanism for the *Timing Service* to provide ETR events (i.e. ETR data and associated Terminal Time timestamp) to subscribed waveforms (as configured via the *TimingService::EtrAccess* interface).

It is assumed that each waveform component that interfaces with the *Timing Service* with the intention of sending data to the *Timing Service* will implement an SCA uses port [2].

B.1.1 Overview

This extension contains as follows:

a. Section B.1, *Introduction*, of this document contains the introductory material regarding the overview, Service Layer Description, Modes, States and Referenced Documents of this document.

b. Section B.2, *Services*, provides summary of service interface uses, interface for each service component, port connections, and sequence diagrams.

c. Section B.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Timing Service* ETR Extension.

d. Section B.4, *IDL*.

e. Section B.5, *UML*, specifies the data types and exceptions that are provided by the *Timing Service* ETR Extension.

f. Appendix B.A – *Abbreviations and Acronyms*.

g. Appendix B.B – *Performance Specification*.

B.1.2 Service Layer Description

B.1.2.1 Timing Service ETR Extension Port Connections

Figure 7 shows the port connections for the *Timing Service* ETR Extension.

Note: All port names are for reference only. Ports identified in black are provided in section A Timing Service.

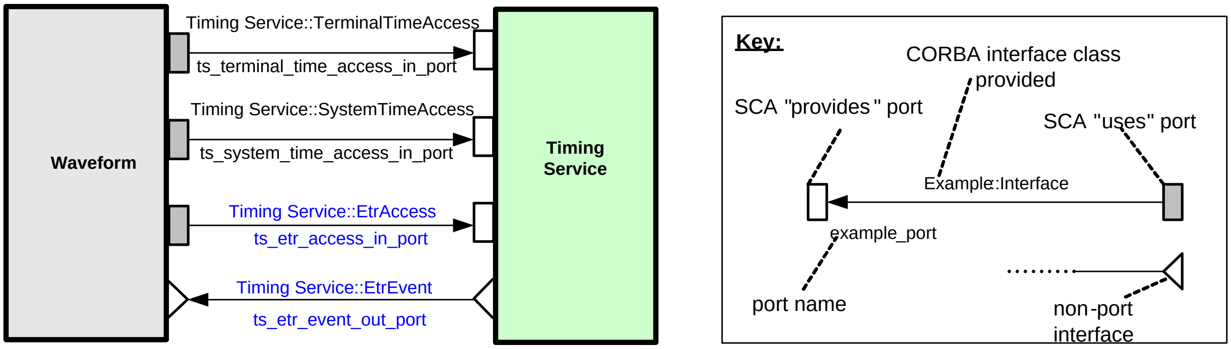

Figure 7 – Timing Service ETR Extension Port Diagram

***Timing Service* ETR Extension Provides Ports Definitions**

ts_etr_access_in_port permits a waveform to enable or disable the output of an ETR event (via the *EtrEvent* interface) when an ETR event occurs.

***Timing Service* ETR Extension Uses Ports Definitions**

ts_etr_event_out_port <non-SCA> permits the *TimingService* to send ETR data to subscribed waveforms.

B.1.2.2 Modes of Service

None

B.1.3 Referenced Documents

There are no changes from the base API (see A.1.3).

B.2 SERVICES

B.2.1 Provide Services

The *Timing Service* ETR Extension provides services with the following service ports, interfaces, and primitives, which can be called by other client components.

Table 5 – Timing Service ETR Extension Provide Service Interface

Service Group (Port Name)	Service (Interface Provided)	Primitives (Provided)
ts_etr_access_in_port	TimingService::EtrAccess	enableEtrEventOutput()
		disableETREventOutput()

B.2.2 Use Services

The *Timing Service* ETR Extension use services consists of the following service ports, interfaces, and primitives.

Table 6 – Timing Service ETR Extension Use Service Interface

Service Group (Port Name)	Service (Interface Provided)	Primitives (Provided)
ts_etr_out_port <non-SCA>	TimingService::EtrEvent	pushEtrEvent()

B.2.3 Interface Modules

B.2.3.1 TimingService

The class diagram for the *External Time Reference Extension* is shown in Figure 8.

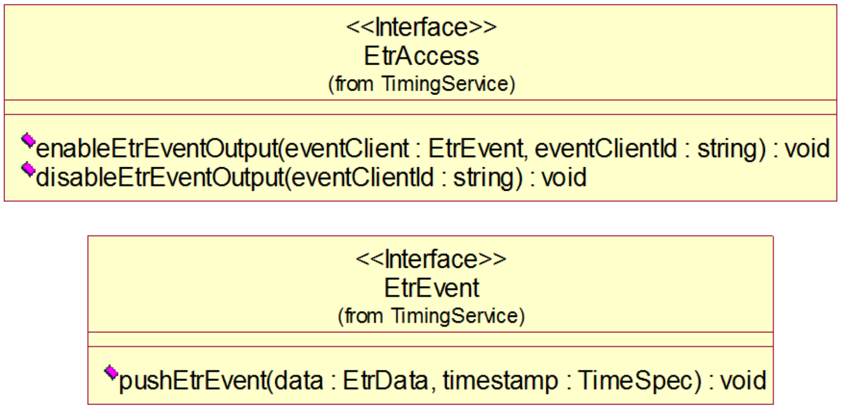
<<Interface>>
EtrAccess
(from TimingService)
enableEtrEventOutput(eventClient : EtrEvent, eventClientId : string) : void
disableEtrEventOutput(eventClientId : string) : void
<<Interface>>
EtrEvent
(from TimingService)
pushEtrEvent(data : EtrData, timestamp : TimeSpec) : void

Figure 8 – External Time Reference Extension Class Diagram

B.2.3.2 EtrAccess Interface Description

The interface design of *EtrAccess* is shown in Figure 9. It provides the mechanism for waveforms to enable or disable the output of an ETR event (via the EtrEvent interface) when an ETR event occurs.

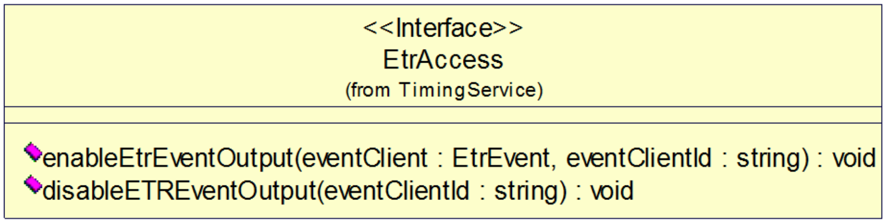
<<Interface>>
EtrAccess
(from TimingService)
enableEtrEventOutput(eventClient : EtrEvent, eventClientId : string) : void
disableETREventOutput(eventClientId : string) : void

Figure 9 – EtrAccess Interface Diagram

B.2.3.3 EtrEvent Interface Description

The interface design of *EtrEvent* is shown in Figure 8. It provides the mechanism for the *Timing Service* to provide ETR events (i.e. ETR data and associated Terminal Time timestamp) to subscribed waveforms (as configured via the *EtrAccess* interface).

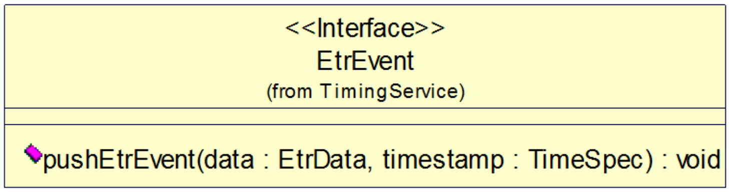

Figure 10 – EtrEvent Interface Diagram

B.2.4 Sequence Diagrams

None

B.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in section B.5 UML. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.

B.3.1 TimingService::EtrAccess

B.3.1.1 *enableEtrEventOutput* Operation

This operation provides the ability to enable the output of an ETR event to the specified client waveform when an ETR event occurs.

B.3.1.2 Synopsis

void enableEtrEventOutput (in EtrEvent eventClient, out string eventClientId);

B.3.1.3 Parameters

Parameter Name	**Description**	**Type**	**Units**	**Valid Range**
eventClient	The CORBA object reference that identifies the client waveform to send ETR Event output to.	EtrEvent (See B.2.3.3)	CORBA Object reference	N/A
eventClientId	The ID given to the client waveform to send ETR event output to.	string	N/A	*Waveform specific*

B.3.1.4 State

This operation is only valid during the ENABLED state.

B.3.1.5 New State

Not Applicable

B.3.1.6 Return Value

None

B.3.1.7 Originator

Waveform

B.3.1.8 Exceptions

None

B.3.1.9 *disableETREventOutput* Operation

This operation provides the ability to disable the output of an ETR event to the specified client waveform.

B.3.1.10 Synopsis

void disableETREventOutput (in string eventClientId);

B.3.1.11 Parameters

Parameter Name	**Description**	**Type**	**Units**	**Valid Range**
eventClientId	The ID that specifies the client waveform to disable output of ETR events to (i.e. the ID retrieved from the *enableEtrEventOutput Operation*).	string	N/A	*Waveform specific*

B.3.1.12 State

This operation is only valid during the ENABLED state.

B.3.1.13 New State

Not applicable

B.3.1.14 Return Value

None

B.3.1.15 Originator

Waveform

B.3.1.16 Exceptions

None

B.3.2 TimingService::EtrEvent

B.3.2.1 *pushEtrEvent* Operation

This operation provides ETR events (i.e. ETR data and associated Terminal Time timestamp) to subscribed waveforms (as configured via the *EtrAccess* interface).

B.3.2.2 Synopsis

void pushEtrEvent (in EtrData data, in TimeSpec timestamp);

B.3.2.3 Parameters

Parameter Name	**Description**	**Type**	**Units**	**Valid Range**
data	A structure containing ETR data (i.e. external reference time at the next local timing pulse and associated TFOM).	EtrData	ETR data	See B.5.4.1 *TimingService::EtrData Structure*
timestamp	The associated Terminal Time corresponding to the time the data reference time was valid. *See Appendix A.E – Terminal Time Timestamp Latency Specification*	TimeSpec	time	See Section A.5.4.1.

B.3.2.4 State

This operation is only valid during the ENABLED state.

B.3.2.5 New State

Not Applicable

B.3.2.6 Return Value

None

B.3.2.7 Originator

Service Provider

B.3.2.8 Exceptions

None

B.4 IDL

B.4.1 TimingServiceEtrExt

```
/*
** TimingServiceEtrExt.idl
*/

#ifndef TIMINGSERVICEETREXT_IDL
#define TIMINGSERVICEETREXT_IDL

#ifndef __TIMINGSERVICE_DEFINED
#include "TimingService.idl"
#endif

module TimingService
{
    struct EtrData
    {
        TimeSpec time;
        unsigned short tfom;
    };

    interface EtrEvent
    {

        void pushEtrEvent (in EtrData data,
                           in TimeSpec timestamp);

    };

    interface EtrAccess
    {

        void enableEtrEventOutput (in EtrEvent eventClient,
                                   out string eventClientId);

        void disableETREventOutput (in string eventClientId);

    };

};

#endif // TIMINGSERVICEETREXT_IDL
```

B.5 UML

This section contains the service component UML diagram and the definitions of all data types referenced (directly or indirectly) by section B.3, Service Primitives and Attributes.

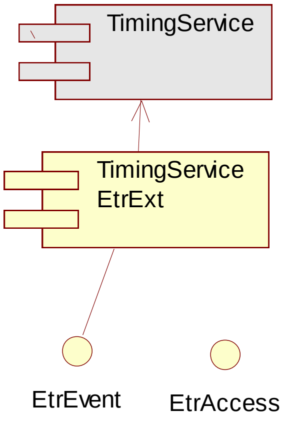

Figure 11 – Timing Service ETR Extension Component Diagram

B.5.1 Data Types

None

B.5.2 Enumerations

None

B.5.3 Exceptions

None

B.5.4 Structures

B.5.4.1 TimingService::EtrData Structure

This structure type defines the ETR data type for the *pushEtrEvent Operation.*

Struct	**Attributes**	**Type**	**Units**	**Valid Range**	**Description**
EtrData	time	TimeSpec	time	N/A	Reference time from epoch 1 Jan 2000 00:00:00 ZULU. See Section A.5.4.1.
	tfom	unsigned short	*Platform dependent*	0 to 65535	TFOM data.

APPENDIX B.A – ABBREVIATIONS AND ACRONYMS

In addition to those defined in the base API (Appendix A.A) this extension defines the following:

ETR	External Time Reference

APPENDIX B.B – PERFORMANCE SPECIFICATION

Table 7 provides a template for the generic performance specification for the *Timing Service* ETR Extension which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 7.

Table 7 – Timing Service ETR Extension Performance Specification

Specification	**Description**	**Units**	**Value**
Worst Case Command Execution Time for ts_etr_access_in_port	*	*	*

Note: (*) These values should be filled in by individual developers.

C. WAVEFORM TIME EXTENSION

C.1 INTRODUCTION

The *Waveform Time Extension* is based upon the *Timing Service* API. It extends the functionality of the *Timing Service* to support the maintenance, storage and recovery of Waveform Time (e.g. after power interruption or power cycle sequences). It retains the methods and attributes defined in the base section A Timing Service API.

The *Waveform Time Extension* defines an SCA provides port [2], *TimingService::WfTimeAccess* interface, as the mechanism for a waveform to provide Waveform Time data samples periodically to the *Timing Service*, request the current Waveform Time data (e.g. after a power down or power interrupt sequence), and clear the stored Waveform Time data.

It is assumed that each waveform component that interfaces with the *Timing Service* with the intention of sending data to the *Timing Service* will implement an SCA uses port [2].

C.1.1 Overview

This extension contains as follows:

a. Section C.1, *Introduction*, of this document contains the introductory material regarding the overview, Service Layer Description, Modes, States and Referenced Documents of this document.

b. Section C.2, *Services*, provides summary of service interface uses, interface for each service component, port connections, and sequence diagrams.

c. Section C.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Timing Service* Waveform Time Extension.

d. Section C.4, *IDL*.

e. Section C.5, *UML*, specifies the data types and exceptions that are provided by the *Timing Service* Waveform Time Extension.

f. Appendix C.A – *Abbreviations and Acronyms*.

g. Appendix C.B – *Performance Specification*.

C.1.2 Service Layer Description

C.1.2.1 Timing Service Waveform Time Extension Port Connections

Figure 12 shows the port connections for the *Timing Service* Waveform Time Extension.

Note: All port names are for reference only. Ports identified in black are provided in section A Timing Service.

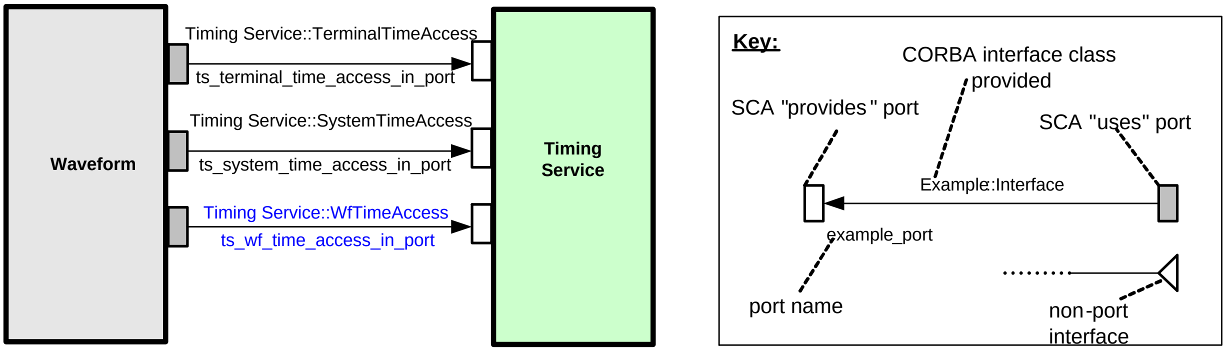
Waveform
Timing Service::TerminalTimeAccess
ts_terminal_time_access_in_port
Timing Service::SystemTimeAccess
ts_system_time_access_in_port
Timing Service::WfTimeAccess
ts_wf_time_access_in_port
Timing Service
Key:
CORBA interface class provided
SCA "provides" port
SCA "uses" port
Example::Interface
example_port
port name
non-port interface

Figure 12 – Timing Service Waveform Time Extension Port Diagram

***Timing Service* Waveform Time Extension Provides Ports Definitions**

ts_wf_time_in_port permits the waveform to provide Waveform Time data samples periodically, to request the current Waveform Time data (e.g. after a power down or power interrupt sequence), and clear the stored Waveform Time data.

***Timing Service* Waveform Time Extension Uses Ports Definitions**

None

C.1.2.2 Modes of Service

None specified

C.1.3 Referenced Documents

There are no changes from the base API (see A.1.3).

C.2 SERVICES

C.2.1 Provide Services

The *Timing Service* Waveform Time Extension provides services consists of the following service ports, interfaces, and primitives, which can be called by other client components.

Table 8 – Timing Service Waveform Time Extension Provide Service Interface

Service Group (Port Name)	Service (Interface Provided)	Primitives (Provided)
ts_wf_time_in_port	TimingService::WfTimeAccess	pushWfTimeData()
		getWfTimeData()
		clearWfTimeData()

C.2.2 Use Services

None

C.2.3 Interface Modules

C.2.3.1 TimingService

The class diagram for the *Waveform Time Extension* is shown in Figure 13.

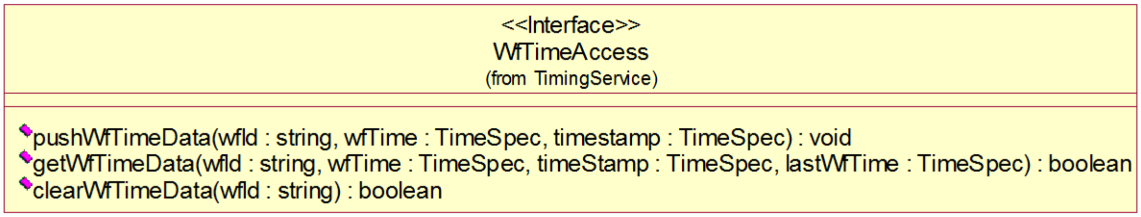
<<Interface>>
WfTimeAccess
(from TimingService)
pushWfTimeData(wfId : string, wfTime : TimeSpec, timestamp : TimeSpec) : void
getWfTimeData(wfId : string, wfTime : TimeSpec, timeStamp : TimeSpec, lastWfTime : TimeSpec) : boolean
clearWfTimeData(wfId : string) : boolean

Figure 13 – Timing Service Waveform Time Extension Class Diagram

C.2.3.2 WfTimeAccess Interface Description

The *WfTimeAccess* interface provides the mechanism for waveforms to provide Waveform Time data samples periodically to the *Timing Service*, to request the Waveform Time (e.g., after a power down or power interrupt sequence), and clear the stored Waveform Time data. In response to a Waveform Time data request, the *Timing Service* computes and returns the current Waveform Time data.

C.2.4 Sequence Diagrams

None

C.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in section C.5 UML. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.

C.3.1 TimingService::WfTimeAccess

C.3.1.1 *pushWfTimeData* Operation

This operation provides the ability for a waveform to provide the *Timing Service* with a Waveform Time data sample (which includes the waveform ID, Waveform Time, and associated Terminal Time timestamp corresponding to the specified waveform).

C.3.1.2 Synopsis

void pushWfTimeData (in string wfId, in TimeSpec wfTime, in TimeSpec timestamp);

C.3.1.3 Parameters

Parameter Name	**Description**	**Type**	**Units**	**Valid Range**
wfId	The unique ID of the calling waveform instance.	string	N/A	*Waveform specific*
wfTime	The Waveform Time provided by the calling waveform.	TimeSpec	time	See Section A.5.4.1.
timestamp	The associated Terminal Time corresponding to time when the wfTime was valid. *See Appendix A.E – Terminal Time Timestamp Latency Specification*	TimeSpec	time	See Section A.5.4.1.

C.3.1.4 State

This operation is only valid during the ENABLED state.

C.3.1.5 New State

Not applicable

C.3.1.6 Return Value

None

C.3.1.7 Originator

Waveform

C.3.1.8 Exceptions

None

C.3.1.9 *getWfTimeData* Operation

This operation provides the ability for a waveform to request the current Waveform Time data (associated with the specified waveform ID).

The *Timing Service* may supplement *wfTime* returned with chronometer time data *(optional)*.

C.3.1.10 Synopsis

boolean getWfTimeData (in string wfId, out TimeSpec wfTime, out TimeSpec timestamp,
out TimeSpec lastWfTime);

C.3.1.11 Parameters

Parameter Name	Description	Type	Units	Valid Range
wfId	The unique ID of the calling waveform instance.	string	N/A	*Waveform specific*
wfTime	The current Waveform Time data computed by *Timing Service*.	TimeSpec	time	See Section A.5.4.1.
timestamp	The Terminal Time corresponding to time when the wfTime was valid. *See Appendix A.E – Terminal Time Timestamp Latency Specification*	TimeSpec	time	See Section A.5.4.1.
lastWfTime	The Last Waveform Time data provided by the calling waveform (via *pushWfTimeData Operation*).	TimeSpec	time	See Section A.5.4.1.

C.3.1.12 State

This operation is only valid during the ENABLED state.

C.3.1.13 New State

Not applicable

C.3.1.14 Return Value

Description	Type	Units	Valid Range
The operation returns TRUE if the Waveform Time data is available (i.e. times returned are valid), FALSE otherwise.	boolean	N/A	TRUE, FALSE

C.3.1.15 Originator

Waveform

C.3.1.16 Exceptions

None

C.3.1.17 *clearWfTimeData* Operation

This operation provides the ability for a waveform to clear its Waveform Time data provided last via the *pushWfTimeData Operation*.

C.3.1.18 Synopsis

boolean clearWfTimeData (in string wfId);

C.3.1.19 Parameters

Parameter Name	**Description**	**Type**	**Units**	**Valid Range**
wfId	The unique ID of the calling waveform instance.	string	N/A	*Waveform specific*

C.3.1.20 State

This operation is only valid during the ENABLED state.

C.3.1.21 New State

Not applicable

C.3.1.22 Return Value

Description	**Type**	**Units**	**Valid Range**
The operation returns TRUE if the Waveform Time data was found and cleared and FALSE otherwise.	boolean	N/A	TRUE, FALSE

C.3.1.23 Originator

Waveform

C.3.1.24 Exceptions

None

C.4 IDL

C.4.1 TimingServiceWfTimeExt

```
/*
** TimingServiceWfTimeExt.idl
*/

#ifndef TIMINGSERVICEWFTIMEEXT_IDL
#define TIMINGSERVICEWFTIMEEXT_IDL

#ifndef __TIMINGSERVICE_DEFINED
#include "TimingService.idl"
#endif

module TimingService
{

    interface WfTimeAccess
    {

        void pushWfTimeData (in string wfId,
                             in TimeSpec wfTime,
                             in TimeSpec timestamp);

        boolean getWfTimeData (in string wfId,
                               out TimeSpec wfTime,
                               out TimeSpec timestamp,
                               out TimeSpec lastWfTime);

        boolean clearWfTimeData (in string wfId);

    };

};

#endif // TIMINGSERVICEWFTIMEEXT_IDL
```

C.5 UML

This section contains the service component UML diagram and the definitions of all data types referenced (directly or indirectly) by section C.3, Service Primitives and Attributes.

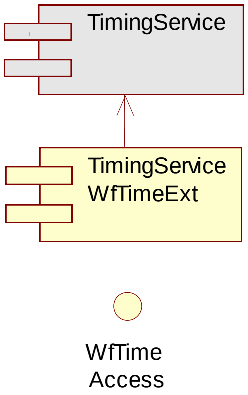

Figure 14 – Timing Service Waveform Time Extension Component Diagram

C.5.1 Data Types

None

C.5.2 Enumerations

None

C.5.3 Exceptions

None

C.5.4 Structures

None

APPENDIX C.A – ABBREVIATIONS AND ACRONYMS

There are no changes from the base API (Appendix A.A).

APPENDIX C.B – PERFORMANCE SPECIFICATION

Table 9 provides a template for the generic performance specification for the *Timing Service* Waveform Time Extension which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 12.

Table 9 – Timing Service ETR Extension Performance Specification

Specification	**Description**	**Units**	**Value**
Worst Case Command Execution Time for ts_wf_time_in_port	*	*	*

Note: (*) These values should be filled in by individual developers.

D. SYSTEM TIME EXTENSION

D.1 INTRODUCTION

The *System Time Extension* is based upon the *Timing Service* API. It extends the functionality of the base *Timing Service, System Time Access* interface, to provide a mechanism for waveforms to set the system time (see section A.2.3.3). It retains the methods and attributes defined in the base section A Timing Service API.

The *System Time Extension* extends the derivation of UTC time to also include network time inputs while utilizing the local timing pulse (see section A Timing Service API).

D.1.1 Overview

This extension contains as follows:

a. Section D.1, *Introduction*, contains the introductory material regarding the Overview, Service Layer Description, Modes, States, and Referenced Documents of this document.

b. Section D.2, *Services* specifies the interfaces, port connections, and sequence diagrams.

c. Section D.3, *Service Primitives and Attributes*, specifies the operations that are provided by the *Timing Service* System Time Extension.

d. SectionD.4, *IDL*.

e. SectionD.5, *UML*.

f. Appendix D.A– *Abbreviations and Acronyms*.

g. Appendix D.B– *Performance Specification*.

D.1.2 Service Layer Description

D.1.2.1 Timing Service System Time Extension Port Connections

Figure 15 shows the port connections for the *Timing Service* System Time Extension.

Note: All port names are for reference only. Ports identified in black are provided in section A Timing Service.

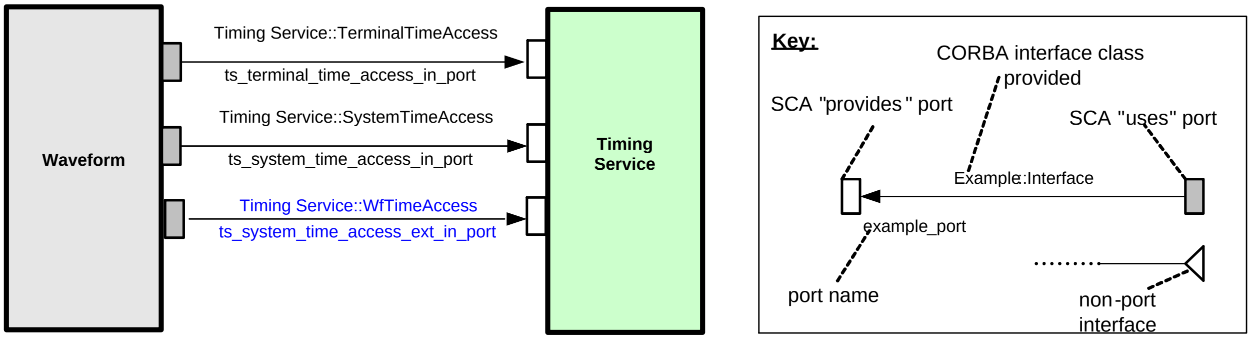
Waveform
Timing Service::TerminalTimeAccess
ts_terminal_time_access_in_port
Timing Service::SystemTimeAccess
ts_system_time_access_in_port
Timing Service::WfTimeAccess
ts_system_time_access_ext_in_port
Timing Service
Key:
CORBA interface class provided
SCA "provides" port
SCA "uses" port
Example:Interface
example_port
port name
non-port interface

Figure 15 – Timing Service System Time Extension Port Diagram

***Timing Service* System Time Extension Provides Ports Definitions**

ts_system_time_access_ext_in_port permits the waveform to set the system time.

***Timing Service* System Time Extension Uses Ports Definitions**

None

D.1.2.2 Modes of Service

None specified.

D.1.3 Referenced Documents

There are no changes from the base API (see A.1.3).

D.2 SERVICES

D.2.1 Provide Services

The *Timing Service* System Time Extension provides services include the following service ports, interfaces, and primitives, which can be called by other client components.

Table 10 – Timing Service System Time Extension Provide Services Interface

Service Group (Port Name)	Service (Interface Provided)	Primitives (Provided)
ts_system_time_access_ext_in_port	TimingService:: SystemTimeAccessExt	setSystemTime()

D.2.2 Use Services

None

D.2.3 Interface Modules

D.2.3.1 TimingService

The class diagram for the *System Time Extension* is shown in Figure 16.

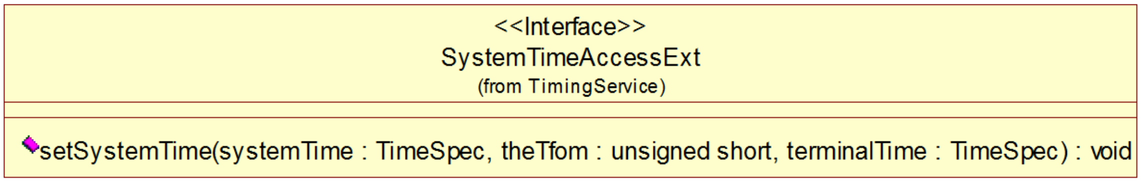
<<Interface>>
SystemTimeAccessExt
(from TimingService)
setSystemTime(systemTime : TimeSpec, theTfom : unsigned short, terminalTime : TimeSpec) : void

Figure 16 – Timing Service System Time Extension Class Diagram

D.2.3.2 SystemTimeAccessExt Interface Description

The *SystemTimeAccessExt* interface provides the mechanism for waveforms to set the system time.

D.2.4 Sequence Diagrams

None

D.3 SERVICE PRIMITIVES AND ATTRIBUTES

To enhance the readability of this API document and to avoid duplication of data, the type definitions of all structured types (i.e., data types, enumerations, exceptions, and structures) used by the Service Primitives and Attributes have been co-located in section D.5 UML. This cross-reference of types also includes any nested structures in the event of a structure of structures or an array of structures.

D.3.1 TimingService::SystemTimeAccessExt

D.3.1.1 *setSystemTime* Operation

This operation allows for a waveform to set the system time.

D.3.1.2 Synopsis

```
void setSystemTime (        in TimingService::TimeSpec systemTime,
                            in unsigned short theTfom,
                            in TimingService::TimeSpec terminalTime )
```

D.3.1.3 Parameters

Parameter Name	Description	Type	Units	Valid Range
systemTime	The new time to use when setting the system time. The System Time value supplied represent the amount of time since epoch 1 Jan 2000 00:00:00 ZULU.	TimeSpec	time	See Section A.5.4.1
theTfom	The TFOM associated with the new system time.	unsigned short	*See Appendix A.D – TFOM and ETE Values for each TFOM numerical value's associated range of Estimated Time Error (ETE).*	1-15
terminalTime	The terminal time when the new system time is effective. This may be equal to what the terminal time was when the new time was supplied.	TimeSpec	time	See Section A.5.4.1

D.3.1.4 State

This operation is only valid during the ENABLED state.

D.3.1.5 New State

Not applicable.

D.3.1.6 Return Value

None

D.3.1.7 Originator

Waveform

D.3.1.8 Exceptions

None

D.4 IDL

D.4.1 TimingServiceSystemTimeExt

```
/*
** TimingServiceSystemTimeExt.idl
*/

#ifndef __TIMINGSERVICESYSTEMTIMEEXT_DEFINED
#define __TIMINGSERVICESYSTEMTIMEEXT_DEFINED

#ifndef __TIMINGSERVICE_DEFINED
#include "TimingService.idl"
#endif

module TimingService {

    interface SystemTimeAccessExt {

        void setSystemTime (
            in TimingService::TimeSpec systemTime,
            in unsigned short theTfom,
            in TimingService::TimeSpec terminalTime
        );
    };
};

#endif
```

D.5 UML

This section contains the service component UML diagram and the definitions of all data types referenced (directly or indirectly) by section D.3, Service Primitives and Attributes.

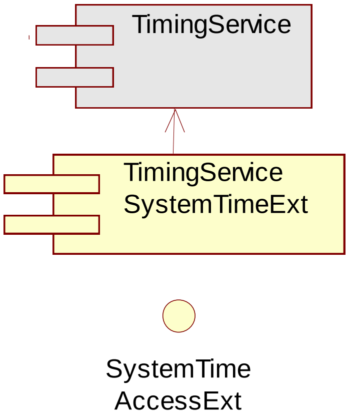

Figure 17 – Timing Service System Time Extension Component Diagram

D.5.1 Data Types

None

D.5.2 Enumerations

None

D.5.3 Exceptions

None

D.5.4 Structures

None

APPENDIX D.A – ABBREVIATIONS AND ACRONYMS

There are no additional abbreviations and acronyms than those defined in the base API (Appendix A.A).

APPENDIX D.B – PERFORMANCE SPECIFICATION

Table 11 provides a template for the generic performance specification for the *Timing Service* System Time Extension which will be documented in the service or device using the interface. This performance specification corresponds to the port diagram in Figure 15.

Table 11 – Timing Service System Time Extension Performance Specification

Specification	Description	Units	Value
Worst Case Command Execution Time for ts_system_time_access_ext_in_port	*	*	*

Note: (*) These values should be filled in by individual developers.