OpenFusion®
CORBA Services
Version 4.1
Time Service

PRISMTECH

OpenFusion®
CORBA Services

TIME SERVICE GUIDE

Part Number: OFCOR-TIMG-41

Doc Issue 09, 13 July 2004

Notices

Copyright Notice

Preface

About the Time Service Guide

The *Time Service Guide* is included with the OpenFusion CORBA Services' *Documentation Set*. The *Time Service Guide* explains how to use the OpenFusion Time Service, which includes both the Timer Event and Time Services.

The *Time Service Guide* is intended to be used with the *System Guide* and other OpenFusion CORBA Services documents included with the product distribution; refer to the *Product Guide* for a complete list of OpenFusion documents.

Intended Audience

The *Time Service Guide* is intended to be used by users and developers who wish to integrate the OpenFusion CORBA Services into products which comply with OMG or J2EE standards for object services. Readers who use this guide should have a good understanding of the relevant programming languages (e.g. Java, IDL) and of the relevant underlying technologies (e.g. J2EE, CORBA).

Organisation

The *Time Service Guide* is organised into two main sections. The first section describes OpenFusion Time Service. This section provides

- a high level description and list of main features
- explanation of the architecture and concepts
- how to use specific features
- detailed explanations of the main interfaces and how to use them
- other information which is needed to use the component

The second section of the *Time Service Guide, Configuration and Management*, provides information on configuring and managing the OpenFusion Time Service using the OpenFusion Graphical Tools. This section includes detailed descriptions of properties specific to the service, plus instructions on how to use the OpenFusion Graphical Tools' Browsers and Managers. It is intended that this section be read in conjunction with the *System Guide*.

Conventions

The conventions listed below are used to guide and assist the reader in understanding the *Time Service Guide*.

Item of special significance or where caution needs to be taken.

Item contains helpful hint or special information.

WIN Information applies to Windows (e.g. NT, 2000) only.

UNIX Information applies to Unix based systems (e.g. Solaris) only.

Hypertext links to WWW and other internet services are shown as *blue italic underlined.*

On-Line (PDF) versions of this document: Items shown as cross references to other parts of the document, e.g., *Contacts* on page vii, behave as hypertext links: readers can jump to that section of the document by clicking on the cross reference.

```
% Commands or input which the user enters on the
  command line of their computer terminal
```

`Courier` fonts indicate programming code and file names.

Extended code fragments are shown in shaded, full width boxes (to allow for standard 80 column wide text), as shown below:

```
NameComponent newName[] = new NameComponent[1];

// set id field to “example” and kind field to an empty string
newName[0] = new NameComponent (“example”, “”);

rootContext.bind (newName, demoObject);
```

Italics and ***Italic Bold*** are used to indicate new terms, or emphasise an item.

Arial Bold is used to indicate user related actions, e.g. **File | Save** from a menu.

Step 1: Indicates that this item is a step or stage of completing a task by a user.

Contacts

PrismTech can be contacted at the following address, phone number, fax and e-mail contact points for information and technical support. Users of the on-line version of this manual can *click* the e-mail addresses below to launch their e-mail client or Web browser to send e-mail direct to PrismTech.

Corporate Headquarters	European Head Office
PrismTech Corporation 6 Lincoln Knoll Lane Suite 100 Burlington, MA 01803 USA	PrismTech Limited PrismTech House 5th Avenue Business Park Gateshead NE11 0NG UK
Tel: +1 781 270 1177 Fax: +1 781 238 1700	Tel: +44 (0)191 497 9900 Fax: +44 (0)191 497 9901

Web: *http://www.prismtechnologies.com*

General Enquiries: *info@prismtechnologies.com*

Support Enquiries: *http://www.prismtechnologies.com/Contacts*

Contents

Table of Contents

Notices iii

Preface v

About the Time Service Guide v

Contacts vii

List of Figures xiii

List of Tables xv

Introduction 1

Time Service 5

1 **Description** 7

1.1 **Overview** 7

Standard OMG Features 7

The Time Service 7

The Timer Event Service 7

1.2 **Architecture and Concepts** 8

Concepts 8

Architecture 8

Time Service 9

Conversions and Comparisons 9

Secure Time 10

Local Time Service 11

Timer Event Service 11

2 **Using Specific Features** 13

2.1 **Time Service** 13

Obtaining the Current Time 13

Operations on Time 14

2.2 **Timer Event Service** 15

3 API Definitions 19
3.1 Interface Definitions 19
4 Supplemental Information 21
4.1 Exceptions. 21
Configuration and Management 23
5 Time Service Configuration 25
5.1 Overview. 25
Common Properties 25
5.2 TimeSingleton Configuration 26
5.3 TimerEventSingleton Configuration 28
5.4 ProcessSingleton Configuration 30
6 Time Service Browser 33
6.1 Overview. 33
6.2 Using the Time Service Browser 33
Properties 34
7 Timer Event Service Browser 35
7.1 Overview. 35
7.2 Using the Timer Event Service Browser 35
Properties 36
Managing Events 37
Set Time 37
Cancel 37
Unregister 37
Index 39

List of Figures

Figure 1 Structure of the Timer Event Service 8
Figure 2 Time Service Browser 34
Figure 3 Timer Event Service Browser 36

List of Tables

Table 1 Possible Overlap Values 10
Table 2 Time Service Interfaces 19
Table 3 Timer Event Service Interfaces 19
Table 4 Time Service Exceptions 21

Introduction

The *OpenFusion Time Service* is one of a range of services and interfaces included with the *OpenFusion CORBA Services* product.

The *OpenFusion Time Service* can be used stand-alone or with other OpenFusion CORBA Services' interfaces and services.

OpenFusion Time Service is standards based, i.e. fully compliant with recognised industry standards and specifications, and supports portability and interoperability.

Time Service

1 Description

1.1 Overview

The OMG Time Service provides a facility to keep a record of time (and an indication of the accuracy of this information) on a distributed CORBA system. The Time Service provides a time source, and a representation of time, to other CORBA and related services, such as the Log Service, the Notification Service, JMS, etc.

Note that the Time Service does not cover the many different time representations available, and does not set out to unify these. Instead, the Time Service's representation of time is based on the *Universal Time Coordinated* (UTC) representation from the X/Open DCE Time Service.

Standard OMG Features

The OpenFusion implementation of the Time Service provides all of the standard features of a OMG-compliant implementation. The OMG Standard Time Service actually comprises two services: the Time Service itself, and the Timer Event Service.

The Time Service

The Time Service provides the following functionality:

- The capability to return the current time, and an indication of its accuracy.
- The capability for administrators of the Time Service to specify whether or not the times generated are secure.
- The time returned is 'monotonic', i.e. it does not run backwards.
- The service is always available and reliable under normal circumstances, and the time it returns falls within an acceptable window of accuracy.

The Timer Event Service

The Timer Event Service provides the following functionality:

- Timer Event Handlers that can be associated with specific Notification Service event channels.
- The Event Handler is used to program time and content into events generated by the event channel.

1.2 Architecture and Concepts

Concepts

The Time Service enables the current time, together with an estimation of its accuracy, to be obtained. The Time Service also provides facilities for manipulating representations of time and time intervals.

The Timer Event Service provides operations for managing time-triggered event handlers and the events they deal with. The event handlers send events to event channels as defined by the Notification Service.

Architecture

The Time Service defines *Universal Time Objects* (UTO) for representing instants in time, and *Time Interval Objects* (TIO) for representing intervals of time. The Time Server provides facilities for creating these objects, and for obtaining the current time. This time value may be obtained from the system clock of the machine on which the Time Server is running, or from a Network Time Protocol (NTP) server.

The Timer Event Service manages Timer Event Handler objects. A client wishing to set up a timed event invokes a registration operation on the service, which then returns a Timer Event Handler. This object delivers events to Notification Service event channels at preprogrammed times. This relationship is shown in *Figure 1*:

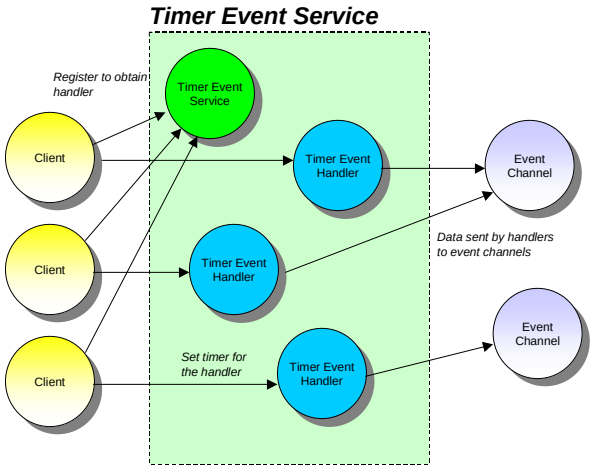


Figure 1 Structure of the Timer Event Service

Time Service

The Time Service unit of time is 100 nanoseconds. Time intervals are expressed as a number of these units. Absolute times are expressed as a number of these units relative to 00:00:00 GMT on 15th October 1582 (which is defined as base time).

Time is represented in a Universal Time Object. This contains an absolute time, an accuracy level associated with that time, and a time-zone.

Time intervals are represented by Time Interval Objects. These contain two absolute times indicating the upper and lower boundaries of the interval.

UTOs and TIOs are created by a Time Server. The Time Server provides facilities for constructing UTOs and TIOs from their component data, and for obtaining a UTO representing the current time.

Structures that contain the data parts of UTOs and TIOs have also been defined. These structures are called `UtcT` and `IntervalT`, respectively.

Conversions and Comparisons

Various operations on Universal Time Objects and Time Interval Objects are defined below.

Time in a UTO is represented as *time* ± *error*, so that it can be converted to an *interval* (*time* - *error*, *time* + *error*). Therefore, an operation `interval()` is provided on a UTO to return an interval and, similarly, `time()` is provided on a TIO to perform the reverse conversion. The interval between two UTOs is generated using the `time_to_interval()` method.

A UTO may be created to contain relative rather than absolute time. In other words, the time value stored in it has a reference point other than 1582/10/15.[1] The method `absolute_time()` may be called on a relative UTO. Doing this returns an absolute UTO obtained by taking the original UTO in relation to the current time.

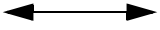
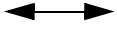
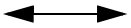
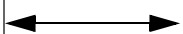
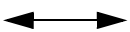
In order to simplify the generation and interpretation of UTOs, a pair of functions are defined in `com.prismt.cos.CosTime.TimeUtil`, namely `UTOtoDate()` and `DateToUTO()`. These functions convert between UTOs and instances of `java.util.Date`, and allow the use of string parsing and other utilities associated with Java `Date`.

1. Note that operations such as `time_to_interval()` give meaningless results when the two operands use different reference points.

The time in one UTO may be compared to another by using the `compare_time()` method. `compare_time()` takes `ComparisonType` as a parameter. The value of `ComparisonType` determines whether the comparison should account for the errors in the UTO (`IntervalC`) or not (`MidC`). The operation returns `TCIndeterminate` if the error ranges of the two UTOs overlap when errors are considered. The operation returns `TCEqual` if the two midpoints are equal when errors are not considered. In either case, `TCGreaterThan` is returned when the time in the object is later than the time in the parameter, or if the value of `TCLessThan` is sooner.

The interval in one TIO may be compared to another using the `overlaps()` method. This returns an `OverlapType` value indicating the type of overlap between the two intervals. A TIO is returned (as an `out` parameter) containing the extent of the overlap - if one exists. The meanings of the possible values of `OverlapType` are shown in the table below.

Table 1 Possible Overlap Values

Overlap Type	OTContainer	OTContained	OTOverlap	OTNoOverlap
Interval in Object				
Interval in Parameter				

The `spans()` method on a TIO is similar to the `overlaps()` method, but takes a UTO as its parameter, obtains the interval represented by the error envelope of that UTO, and makes the comparison against that interval. In fact, `A.spans(B, C)` is equivalent to `A.overlaps(B.interval(), C)` - where `A` and `C` are TIOs, and `B` is a UTO.

Secure Time

The Time Server may be able to return a *secure* time, which indicates that various precautions have been taken to ensure that the server always has the right time. The OpenFusion Time Service can be configured to use either the system clock of the machine it is running on, or an NTP server as its time source. The Time Service should be configured to return either secure or insecure time according to the security of the time source used. The Administration Manager is used to specify a secure time source.

The criteria to be followed for secure time are specified in the *Common Object Services Specification*. Briefly, time management operations such as setting the current time, and designating authoritative time sources, must be audited by, and restricted to, authorised administrators. Time must also be synchronised across the network in a secure manner.

Local Time Service

It may be acceptable, and more convenient, in some circumstances to be able to access some of the Time Service functionality without having to configure and run a Time Server. The `com.prismt.cos.CosTime.TimeUtil` class provides a number of locally executing, static methods equivalent to those in the Time Service:

- *`local_universal_time ()`*: returns a `UTO` representing the current time, and throws the `TimeUnavailable` exception when NTP is configured and the server is not available.
- *`local_new_interval (long lower_bound, long upper_bound)`*: creates a new `TIO` object from its components.
- *`local_new_universal_time (long mean, long error, short tdf)`*: creates a new `UTO` object from its components.
- *`local_uto_from_utc (UtcT utc)`*: creates a new `UTO` object from a `utc`.
- *`timestamp ()`*: returns a `UtcT` representing the current time. The local machine's clock is used when NTP is configured but the server is unavailable.

Timer Event Service

The Timer Event Service manages Timer Event Handler objects. When a timer event handler is created, it is associated with an event channel which is the destination for all the events sent by that handler. The event channel associated with a handler cannot be changed.

The timer governing when the event is sent by the handler to the event channel can be set using one of three timer types:

- *TTAbsolute*: The UTO provided is an absolute time expressed in the units used by the time service. Note that setting the timer with an expired absolute time results in a CORBA system exception.
- *TTRelative*: The UTO provided is a relative time expressed in the units used by the time service. Setting the timer with a relative time of zero results in the previous time setting being used again, when the previous type was also of type `TTRelative`.

- *TTPeriodic*: The UTO provided is a periodic time expressed in the units used by the time service. The event is triggered repeatedly at intervals of the periodic time until the timer is reset or cancelled.

The status of a timer event handler has one of four possible values associated with it, depending on its current state. The definition of these values in the OMG's Time Service V1.0 specification is somewhat ambiguous. The following is PrismTech's interpretation of the specification which is used in the OpenFusion implementation:

- *ESTimeCleared*: An event is not set to be triggered. Either the timer was cancelled before the previously set triggering time occurred, or the handler has just been created and the time has yet to be set.
- *ESTimeSet*: An event has been set with a time in the future. The event will trigger when that time is reached. Note that when the timer is set with periodic time, this value indicates that the previous event was successfully triggered, or that the first event has not yet been triggered.
- *ESTriggered*: The event has already been triggered and the data has been sent over the event channel. Note that when the event is set with periodic time, this value never occurs.
- *ESFailedTrigger*: The time set for an event to be triggered has been reached but the data could not be delivered over the event channel. This state is set when the *push* to the event channel throws a `Disconnected` exception, or the event channel cannot be contacted. Note that when the event is set with periodic time, this value will last only until the event is ready to be sent again.

The data associated with a Timer Event Handler can be changed at any time and the new data is sent to the event channel at all future triggering times. When an event is sent to the event channel, the data also includes a timestamp of the actual event time.

2 Using Specific Features

This chapter describes how, with the use of simple examples, to use specific features of the Time Service. Additional example applications using the service, complete with source code and descriptions of how to compile and run them, are supplied separately as part of the product distribution. Topics covered here include:

- Time Service
- Timer Event Service

Note

- There is little or no error-checking in the examples shown here. Code to deal with exceptions has generally been omitted for the sake of clarity and brevity. These exceptions must of course be properly caught and handled in a working system.
- The following libraries must be imported into any application using the OpenFusion Time and Timer Event Services:

```
import org.omg.TimeBase.*;
import org.omg.CosTime.*;
import org.omg.CosTimerEvent.*;
```

2.1 Time Service

This section is sub-divided into the following topics:

- *Obtaining the Current Time*: Shows how to resolve the Time Server, obtain the current time from it, and extract the component parts of that time value.
- *Operations on Time*: Illustrates the use of the operations that compare and convert UTOs and TIOs.

Obtaining the Current Time

This section provides a guide to the Time Service. It contains annotated example code which illustrates the use of the service.

A time server must first be located. Here, the `resolve_initial_references` operation obtains a time server reference:

```
TimeService server = null;
try
{
   server = TimeServiceHelper.narrow
      (orb.resolve_initial_references (“TimeService”));
}
catch (org.omg.CORBA.ORBPackage.InvalidName ex)
{
   System.err.println (“Failed to resolve time server”);
```

```
   System.exit (1);
}
```

Next, the current time is obtained:

```
UTO timenow = null;
try
{
   timenow = server.universal_time ();
}
catch (TimeUnavailable ex)
{
   System.err.println (“Time unavailable.”);
}
```

Its value is then displayed:

```
if (timenow != null)
{
   System.out.println
   (
      “Time now is “ + timenow.time () + “, Error is “ +
      timenow.inaccuracy () + “, TZ is “ + timenow.tdf () + “ mins”
   );
   System.out.println
   (
      “(“ +
      com.prismt.cos.CosTime.TimeUtil.UTOtoDate (timenow).toString () +
      “)”
   );
}
```

Operations on Time

The following code runs for a specified number of seconds:

```
public void timed_work (int seconds)
{
   TIOHolder dummy = new TIOHolder ();

   // create a relative time with the required duration

   UTO offset = server.new_universal_time (seconds * 10000000, 0, (short) 0);

   // convert relative to absolute time

   UTO finish_time = offset.absolute_time ();

   try
   {
      // make an interval representing the time during which we should work

      TIO working_interval =
         server.universal_time ().time_to_interval (finish_time);

      // loop while the time is still enclosed in that interval

      while (working_interval.spans (server.universal_time (), dummy)
             == OverlapType.OTContainer)
```

```
      {
         do_some_work ();
      }
   }
   catch (TimeUnavailable ex)
   {
      System.err.println ("Time unavailable.");
      return;
   }
}
```

This could also be accomplished without using an interval as follows:

```
// create a relative time with the required duration

UTO offset = server.new_universal_time (seconds * 10000000, 0, (short) 0);

// convert relative to absolute time

UTO finish_time = offset.absolute_time ();
try
{
   // loop while finish_time > current time

   while (finish_time.compare_time
            (ComparisonType.MidC, server.universal_time ())
          == TimeComparison.TCGreaterThan)
   {
      do_some_work ();
   }
}
catch (TimeUnavailable ex)
{
   System.err.println ("Time unavailable.");
   return;
}
```

2.2 Timer Event Service

The Timer Event Service creates and manages Timer Event Handlers that hold information about an event to be triggered at a given time, and the action that is to be taken when that event is triggered. Events consist of data stored in an Any and are sent by the service to event channels using *push semantics*. Timer Event Handlers are obtained by registering an event channel and associated data with the service. The register operation returns a reference to a Timer Event Handler.

```
TimerEventHandler handler = server.register (consumer, any);
```

The data associated with the Timer Event Handler can be changed after registration; only the event channel associated with the handler cannot be modified.

The handler must have its timer set in order for events to be triggered. The following code segments obtain the current time from the Time Service and add five seconds to create an absolute time five seconds in the future. A unit of time in the Time Service is 100 nanoseconds. Therefore there are 10,000 units of time in a millisecond.

```
UTO timenow = null;
try
{
   timenow = timeServer.universal_time ();
}
catch (TimeUnavailable ex)
{
   ex.printStackTrace ();
}
```

```
UTO newtime = timeServer.new_universal_time
(
   timenow.time () + (5000 * 10000),
   timenow.inaccuracy (),
   timenow.tdf ()
);
```

The calculated time and type of time must be set in the timer. In this case we have calculated a new absolute time:

```
TimeType ttype = TimeType.TTAbsolute;

handler.set_timer (ttype, newtime);
```

The data associated with an event can be changed at any time. The new data will be sent the next time the event is triggered. The timer does not necessarily have to be reset once the data has been changed. For example, once a periodic event is set up, the data can be changed as often as is required with the result that the up-to-date data will be sent with each future event.

```
handler.set_data (any);
```

The `time_set` operation of the event handler returns `true` when the time has been set for an event that has yet to be triggered, and `false` otherwise. This operation also returns the current value of the timer in the `out` parameter. The status attribute of the handler returns the *EventStatus* that reflects the current state of the handler.

```
UTOHolder utoh = new UTOHolder ();
System.out.println ("Time Set? " + handler.time_set (utoh));
System.out.println ("Status? " + handler.status ());
```

The data is delivered using the `push` method of the event channel when an event is triggered. The event that is delivered is wrapped into a *TimerEventT* structure with two fields. The first field, `utc`, contains the actual time at which the event was

triggered, while the second field, `event_data`, contains the data. The `event_time` method of the Timer Event Service is used to obtain a UTO containing the time at which the event was triggered.

```
public void push (org.omg.CORBA.Any data) throws Disconnected
{
   if (!connected)
   {
      throw new Disconnected ();
   }
   TimerEventT value = TimerEventTHelper.extract (data);
   UTO uto = server.event_time (value);
   System.out.println ("Received event: " + value);
}
```

The `cancel_timer` operation cancels a timer that has yet to be triggered. This method returns `true` when an event is actually cancelled, and `false` otherwise. After cancelling an event, the status of the timer becomes `ESTimeCleared`.

```
boolean b = handler.cancel_timer ();
```

The `unregister` operation is used to notify the Timer Event Service that a Timer Event Handler is not going to be used again. Any subsequent attempt to use the handler raises the `CORBA::OBJECT_NOT_EXIST` exception.

```
server.unregister (handler);
```

The Timer Event Service periodically checks whether any registered Timer Event Handlers are associated with Push Consumers that no longer exist. The handler will be automatically unregistered when it is found to be associated with an inactive and non-persistent consumer. This check takes place every five minutes.

3 API Definitions

The complete IDL API is provided elsewhere as part of the product distribution.

3.1 Interface Definitions

The Time Service defines three interfaces: a factory interface (`TimeService`), an interface for representing a single time (`UTO`), and an interface for representing a time interval (`TIO`). These are summarised in *Table 2*:

Table 2 Time Service Interfaces

Interface	Purpose
TimeService	A factory interface for UTOs and TIOs. Used to create TIOs and UTOs from component parts, and UTOs from the current time.
UTO	An interface representing an instant of time. Holds a time value along with an accuracy estimate and a time-zone.
TIO	An interface representing an interval of time. It contains two time values.

The Timer Event Service defines two interfaces for managing timed events. These are summarized in *Table 3*:

Table 3 Timer Event Service Interfaces

Interface	Purpose
TimerEventService	Register and unregister `TimerEventHandler` objects. The event (proxy) consumer to which the timed events are delivered is fixed at registration time.
TimerEventHandler	An object that generates events at preset times. The content and timing of the events may be set and reset at any time.

4 Supplemental Information

4.1 Exceptions

The Time Service can throw only one exception as detailed in *Table 4*:

Table 4 Time Service Exceptions

Exception	Description
TimeUnavailable	Thrown by `TimeService::secure_universal_time()` when a secure time is not available. It is also thrown by `TimeService::universal_time()` to indicate that no time source is available, e.g. when an NTP server does not respond.

The Timer Event Service does not throw any service-specific exceptions.

Configuration and Management

5 Time Service Configuration

5.1 Overview

The configuration of Singleton properties specific to the Time Service is described in this section. These properties appear in the Administration Manager, a graphical user interface (GUI) based administration tool included with the OpenFusion Graphical Tools.

The Administration Manager can be used to set the Singleton properties. These properties can also be set programatically, generally as described in the service description sections.

Details for configuring Persistence, Logging, CORBA, Java and System properties for the Time Service are described in the *System Guide*.

Common Properties

Instances of some common properties are used by a number of different OpenFusion CORBA Services' interfaces and services. Settings for these property instances appear in the Administration Manager's Object Hierarchy for the service's Singleton node. This small group of properties are included in this section in order to facilitate configuration of the service while using the Administration Manager. These properties include:

- IOR Name Service Entry
- IOR URL
- IOR File Name
- Resolve Name
- IOR Name Service

5.2 TimeSingleton Configuration

IOR Name Service Entry

The Naming Service entry for the Singleton.

Property Name	`Object.Name`
Property Type	`FIXED`
Data Type	`STRING`
Accessibility	`READ/WRITE`
Mandatory	`NO`

IOR URL

The *IOR URL* property specifies the location of an Interoperable Object Reference (IOR) for the Service, using the Universal Resource Locator (URL) format. This information is used when a client attempts to resolve a reference to the Service. Some examples are:

```
file:/usr/users/openfusion/servers/TimeService.ior
http://www.prismtech.com/of/servers/TimeService.ior
corbaloc::server.prismtechnologies.com/TimeService
```

OpenFusion supports URLs in *Corbaloc*, *Corbaname*, *file*, *FTP* and *HTTP* URL formats, although some ORBs do not support all of these mechanisms. Consult your ORB documentation for specific details.

Property Name	`IOR.URL`
Property Type	`FIXED`
Data Type	`URL`
Accessibility	`READ/WRITE`
Mandatory	`NO`

IOR File Name

The *IOR File Name* option specifies the name and location of the IOR file for the Singleton. If this property is not set, the IOR file name will be:

```
<INSTALL>/domains/<domain>/<node>/<service>/<singleton>/<singleton>.ior
```

where <INSTALL> is the OpenFusion installation path. See the *System Guide* for details of the domains directory structure.

Property Name	IOR.File
Property Type	FIXED
Data Type	FILE
Accessibility	READ/WRITE
Mandatory	NO

Resolve Name

The ORB Service resolution name used to resolve calls to the Singleton

Property Name	ResolveName
Property Type	FIXED
Data Type	STRING
Accessibility	READ/WRITE
Mandatory	YES

IOR Name Service

The name of the Naming Service which will be used to resolve the Singleton object.

Property Name	IOR.Server
Property Type	FIXED
Data Type	STRING
Accessibility	READ/WRITE
Mandatory	NO

NTP Server

The *NTP Server* property specifies the name of the host running an NTP server from which the Time Service will obtain the time when *Time Source* is set to NTP. The local host is used when this is left blank.

Property Name	NTPServer
Property Type	STATIC

Data Type	STRING
Accessibility	READ/WRITE
Mandatory	YES

Time Source

The *Time Source* option determines whether the Time Service uses the local system clock (*Local*) or a network time server (*NTP*) as its time source.

Property Name	Source
Property Type	STATIC
Data Type	ENUM
Accessibility	READ/WRITE
Mandatory	YES

Service is Secure

The *Time is Secure* option can be selected, under the control of a system administrator, when there is a secure time policy in place for the distributed system. It enables the use of secure time operations which conform to the OMG secure time criteria.

Property Name	Secure
Property Type	STATIC
Data Type	BOOLEAN
Accessibility	READ/WRITE
Mandatory	YES

5.3 TimerEventSingleton Configuration

IOR Name Service Entry

The Naming Service entry for the Singleton.

Property Name	Object.Name
Property Type	FIXED
Data Type	STRING
Accessibility	READ/WRITE
Mandatory	NO

IOR URL

The *IOR URL* property specifies the location of an Interoperable Object Reference (IOR) for the Service, using the Universal Resource Locator (URL) format. This information is used when a client attempts to resolve a reference to the Service. Currently only *http* and *file* URLs are supported, for example:

```
file:/usr/users/openfusion/TimerEventSingleton.ior
http://www.prismtechnologies.com/openfusion/TimerEventSingleton.ior
```

Property Name	IOR.URL
Property Type	FIXED
Data Type	URL
Accessibility	READ/WRITE
Mandatory	NO

IOR File Name

The *IOR File Name* option specifies the name and location of the IOR file for the Singleton. If this property is not set, the IOR file name will be:

```
<INSTALL>/domains/<domain>/<node>/<service>/<singleton>/<singleton>.ior
```

where <INSTALL> is the OpenFusion installation path. See the *System Guide* for details of the domains directory structure.

Property Name	IOR.File
Property Type	FIXED
Data Type	FILE
Accessibility	READ/WRITE
Mandatory	NO

Resolve Name

The ORB Service resolution name used to resolve calls to the Singleton

Property Name	ResolveName
Property Type	FIXED
Data Type	STRING
Accessibility	READ/WRITE
Mandatory	YES

IOR Name Service

The name of the Naming Service which will be used to resolve the Singleton object.

Property Name	IOR.Server
Property Type	FIXED
Data Type	STRING
Accessibility	READ/WRITE
Mandatory	NO

5.4 ProcessSingleton Configuration

IOR Name Service Entry

The Naming Service entry for the Singleton.

Property Name	Object.Name
Property Type	FIXED
Data Type	STRING
Accessibility	READ/WRITE
Mandatory	NO

IOR URL

The *IOR URL* property specifies the location of an Interoperable Object Reference (IOR) for the Service, using the Universal Resource Locator (URL) format. This information is used when a client attempts to resolve a reference to the Service. Currently only *http* and *file* URLs are supported, for example:

```
file:/usr/users/openfusion/servers/TradingService.ior
http://www.prismtech.com/openfusion/servers/NameService.ior
```

Property Name	IOR.URL
Property Type	FIXED
Data Type	URL
Accessibility	READ/WRITE
Mandatory	NO

IOR File Name

The *IOR File Name* option specifies the name and location of the IOR file for the Singleton. If this property is not set, the IOR file name will be:

```
<INSTALL>/domains/<domain>/<node>/<service>/<singleton>/<singleton>.ior
```

where `<INSTALL>` is the OpenFusion installation path. See the *System Guide* for details of the `domains` directory structure.

Property Name	`IOR.File`
Property Type	`FIXED`
Data Type	`FILE`
Accessibility	`READ/WRITE`
Mandatory	`NO`

IOR Name Service

The name of the Naming Service which will be used to resolve the Singleton object.

Property Name	`IOR.Server`
Property Type	`FIXED`
Data Type	`STRING`
Accessibility	`READ/WRITE`
Mandatory	`NO`

6 Time Service Browser

6.1 Overview

The Time Service Browser, supplied as part of the OpenFusion implementation of the CORBA Time Service, is a tool for viewing the time from a COS Time Server or a Universal Time Object.

6.2 Using the Time Service Browser

The Time Service Browser can only be started if the Time Service has been started. To start the Time Service Browser, right-click on a running *TimeSingleton* in the *Object Hierarchy* and select **Time Service Browser** from the pop-up menu. See the *System Guide* for details.

Alternatively, start the Time Service Browser from the command line with the following command:

```
%  run com.prismt.cos.browser.time.TimeBrowser
   -name TimeService
```

The Time Service Browser displays the time from the Time Server or Universal Time Object. The time is shown as text and also in graphical form, as illustrated in *Figure 2*. The time display is in terms of the local time zone and is updated every second. The display is re-synchronised with the server once per minute.

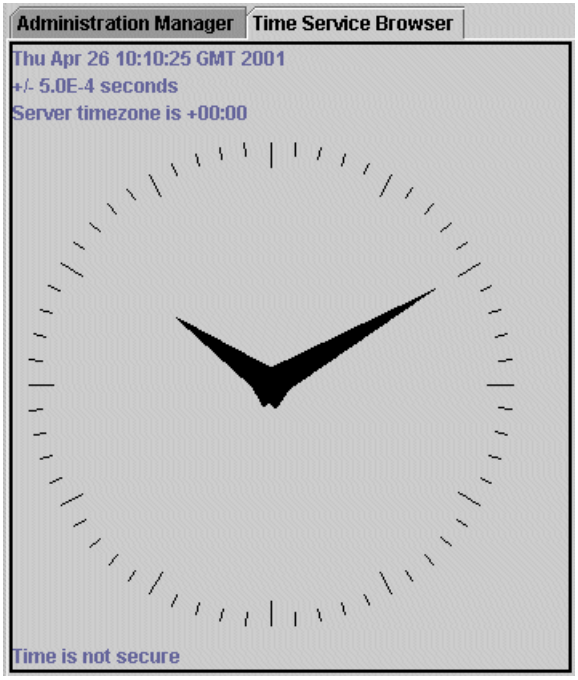


Figure 2 Time Service Browser

Properties

The following time properties are displayed:

- The claimed accuracy of the server time.
- The time zone of the server.

The status line at the bottom left of the browser gives information about the object being browsed. The status will be one of the following:

- *Time is secure*: The object being browsed is a Time Server which returns a secure time.
- *Time is not secure*: The object being browsed is a Time Server which does not return a secure time.
- *Time is a UTO*: The object being browsed is a Universal Time Object.
- *Time is unavailable*: The object being browsed is a Time Server which is either unreachable or cannot provide time.
- *No object selected*: No object has been selected for browsing.

7 Timer Event Service Browser

7.1 Overview

The COS Timer Event Service is an optional service within the COS Time Service. The Timer Event Browser allows timer event handlers associated with the COS Timer Event Service to be viewed and changed. The tasks that can be performed from this browser include:

- Viewing the state of the timer event handlers.
- Setting the trigger time for timed events.
- Cancelling timed events.
- Unregistering timer event handlers.

7.2 Using the Timer Event Service Browser

The Timer Event Service Browser can only be started if the Time Service has been started. To start the Timer Event Service Browser, right-click on a running *TimerEventSingleton* in the *Object Hierarchy* and select **Timer Event Service Browser** from the pop-up menu. See the *System Guide* for details.

Alternatively, start the Timer Event Service Browser from the command line with one of the following commands:

```
% run com.prismt.cos.browser.time.TimerEventBrowser
  -name TimerEventService
```

```
% run com.prismt.cos.browser.time.TimerEventBrowser
  -name TimerEventService -url file:<XML_File_URL>
```

Where `<XML_File_URL>` is the URL of the domain XML file, which defaults to `<INSTALL>/domains/OpenFusion/OpenFusion.xml` (where `<INSTALL>` is the OpenFusion installation directory).

The Timer Event Service Browser shows the status of all the timer event handlers within the timer event service, as shown in *Figure 3*.

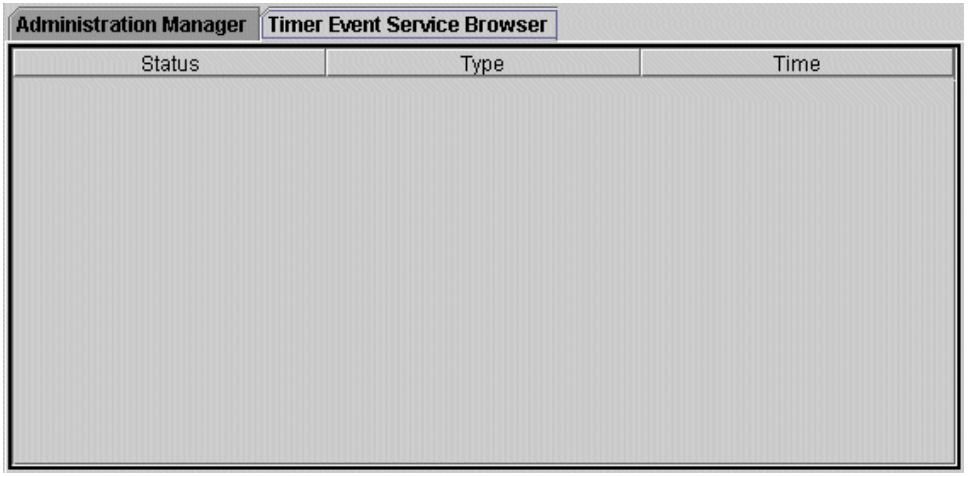


Figure 3 Timer Event Service Browser

Properties

The following properties are displayed for each timer event handler.

Status

The status of a timer event handler can be:

- *Cleared*: The timer has been cancelled.
- *Triggered*: The trigger time for the event has been reached and the event has been delivered.
- *Pending*: The trigger time for the event has not yet been reached.
- *Failed to trigger*: The trigger time for the event has been reached but the handler was unable to deliver the event.

Type

The way in which the time for a timer event is specified. This will be one of the following:

- *Absolute*: The time shown is the time for which the event is set.
- *Relative*: the time shown is the amount of time between when the timer was set and when it will trigger.
- *Periodic*: The event will trigger at intervals of the time shown.

Time

The time for the timer event, specified in accordance with the *Type* property.

Managing Events

The following operations in the Timer Event Browser can be performed on individual timer events or on a selection. Use Shift+Click to extend a selection or Ctrl+Click to add a single event to a selection.

Set Time

Right-click on an event or selection of events and choose *Set Time* from the pop-up menu to change the trigger time of the selected events.

The required time format may vary according to your locale. An error message which includes an example formatted time will be shown when the program is unable to parse the input.

Cancel

Right-click on an event or selection of events and choose *Cancel* from the pop-up menu to cancel the selected events. This will change the status of the timer event handler to *Cleared* and the events will not fire. However, the content and destinations of the events remain set and may be reactivated by setting a new value for the trigger time.

Unregister

Right-click on an event or selection of events and choose *Unregister* from the pop-up menu to unregister the selected event handlers. They will be removed from the timer event service and will cease to fire any further events.

Index

Index

A

API Definitions 19

C

Cancel Option 37

E

Examples
  Time Service 13, 14
Timer Event Service 15
Exceptions, Time Service 21

I

Interfaces, Time Service 19
IOR File Name (property) 26, 29, 31
IOR Name Service (property) 27, 30, 31
IOR Name Service Entry (property) 26, 28, 30
IOR URL (property) 26, 29, 30
IOR.File (property) 27, 29, 31
IOR.URL (property) 26, 29, 30

N

Network Time Protocol (NTP) 8
NTP Server (property) 27
NTPServer (property) 27

O

Object.Name (property) 26, 28, 30
Obtaining the Current Time 13

P

ProcessSingleton Configuration 30

R

Resolve Name (property) 27, 29
ResolveName (property) 27, 29

S

Secure (property) 28
Service is Secure (property) 28
Set Time 37
Singletons
  TimeSingleton 26
Source (property) 28

Starting
Time Service Browser 33

T

Time
Obtaining Current 13
Operations 14
Secure 10
Time Interval Object (TIO) 8
Time Service
Architecture 8
Browser 33
Concepts 9
Configuration 25
Example
Time, Obtaining Current 13
Time, Operations 14
Exceptions 21
Interface Definitions 19
Local 11
Overview 7
Service Features 7
Time Interval Object (TIO) 8
Time, Secure 10
Universal Time Object (UTO) 8
Using Service Features 13
Time Source (property) 28
Timer Event Manager
Cancel Option 37
Set Time Option 37
Unregister Option 37
Timer Event Service
Architecture 8
Browser 35
Description 11
Example Usage 15
Features 7
TimerEventSingleton Configuration 28
TimeSingleton Configuration 26

U

Universal Time Object (UTO) 8
Unregister Option 37