The OXF Model

Project: oxf

Table of Contents

Model Overview

1. Purpose and current status of model
This model describes the requirements analysis and design of the Core Object eXecution Framework (COXF).

2. The structure of the model
The model has an Analysis package that specifies the requirements, use-cases, actors, analysis classes and analysis scenarios.
The Design package specifies the architecture and design scenarios derived from the Analysis package.

3. How to browse the model?
Analysis:
- Start with the FunctionalDomains diagram under the Analysis package. Every package is traced to a set of high-level requirements (also directly under analysis). Each package has a main diagram that is a use case diagram and can be opened using the Open Main Diagram command.
- Each analysis package has a Use Case diagram called “Requirements Vs. UseCases” which details how the use cases are traced to the requirements. Also, there is an Object Model Diagram called Actors which details the actors involved in the package.
- You can navigate from the use cases to their referenced sequence diagrams
Design:
- Start with the Package overview OMD and advance in the same technique as in the analysis.
- The core behavioral framework interfaces and the design level scenarios are located in the CoreAPI package (in Design::oxf::Core).
- The default implementation of the interfaces is located in the CoreImplementation package (in Design::oxf::Core).

Package Information

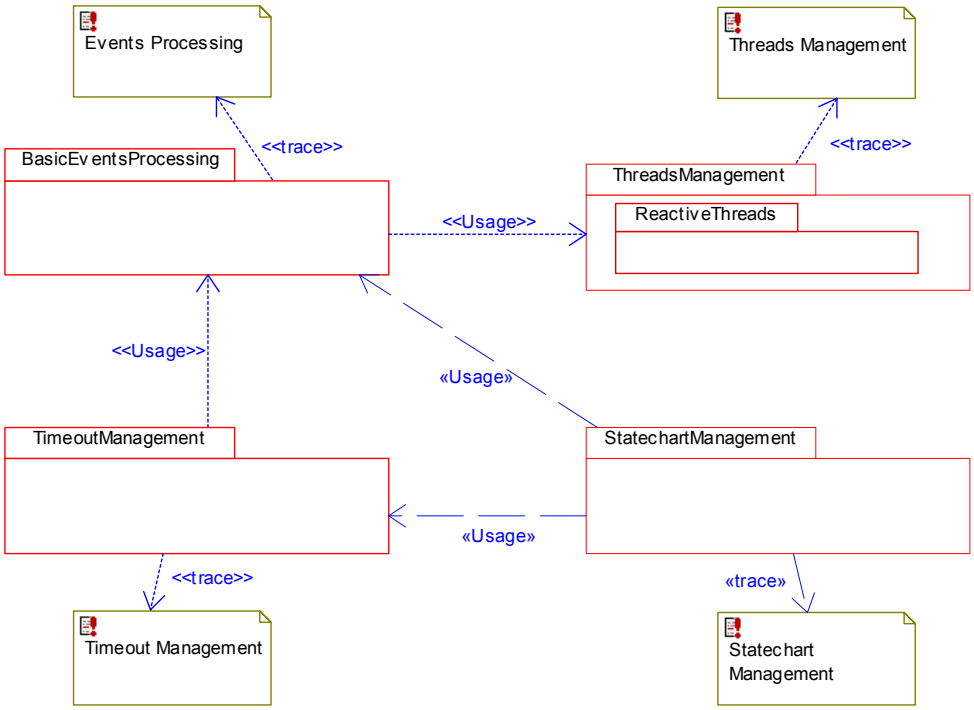
Package: Analysis

Description: The core behavioral framework analysis

Object Model Diagram Information

Object Model Diagram name: FunctionalDomains

Description: The analysis domains



Actor Information for Package: Analysis

Actor name: Application Object

Description:A generic application object.

Relation information for Actor Application Object

Relation name: itsInitialize event pump

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsInitialize event pump
LinkName:
RoleName: itsInitialize event pump
Type: Association
Description:

Relation name: itsStart control flow of thread

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsStart control flow of thread
LinkName:
RoleName: itsStart control flow of thread
Type: Association
Description:

Relation name: itsTerminate control flow of thread

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsTerminate control flow of thread
LinkName:
RoleName: itsTerminate control flow of thread
Type: Association
Description:

Relation name: itsCreate an active reactive object

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsCreate an active reactive object
LinkName:
RoleName: itsCreate an active reactive object
Type: Association
Description:

Relation name: itsInitialize an active reactive object

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsInitialize an active reactive object
LinkName:
RoleName: itsInitialize an active reactive object
Type: Association
Description:

Relation name: itsTerminate an active reactive object

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsTerminate an active reactive object
LinkName:
RoleName: itsTerminate an active reactive object
Type: Association
Description:

Name	Inverse	Source	Target
itsInitialize event pump		Application Object	Initialize event pump
itsStart control flow of thread		Application Object	start control flow of thread
itsTerminate control flow of thread		Application Object	terminate control flow of thread
itsCreate an active reactive object		Application Object	Create an active reactive object
itsInitialize an active reactive object		Application Object	Initialize an active reactive object
itsTerminate an active		Application Object	Terminate an active

reactive object			reactive object

Package Information

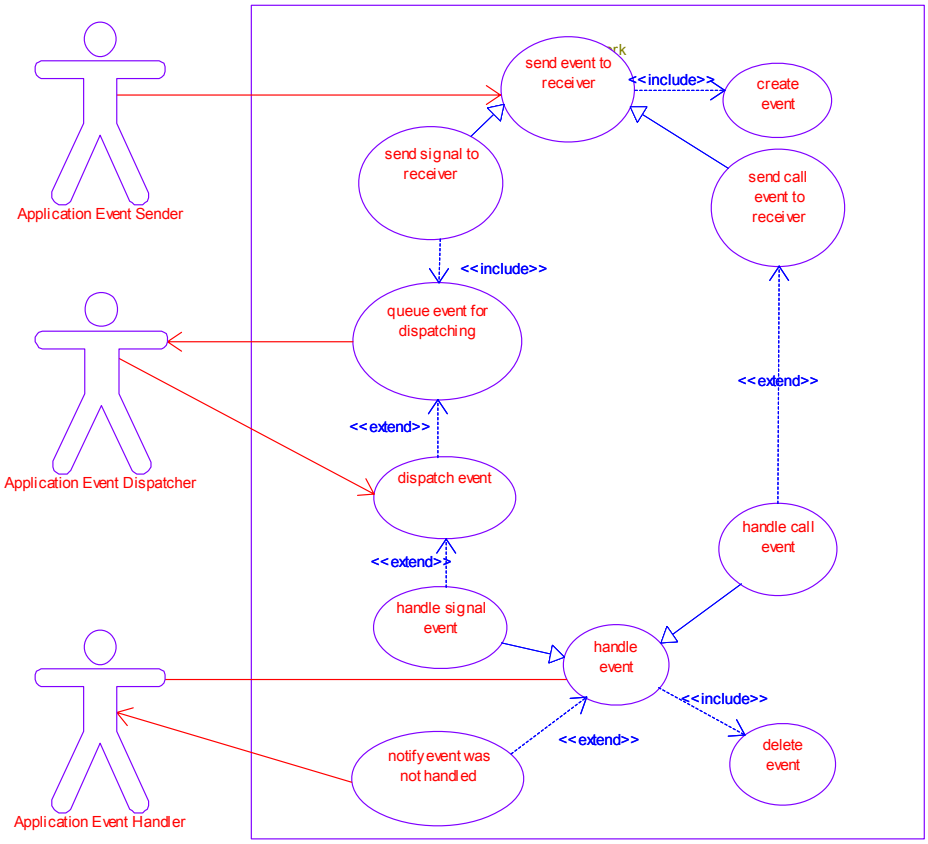
Description: The core behavioral framework analysis

Package: BasicEventsProcessing

Use Case Diagram Information

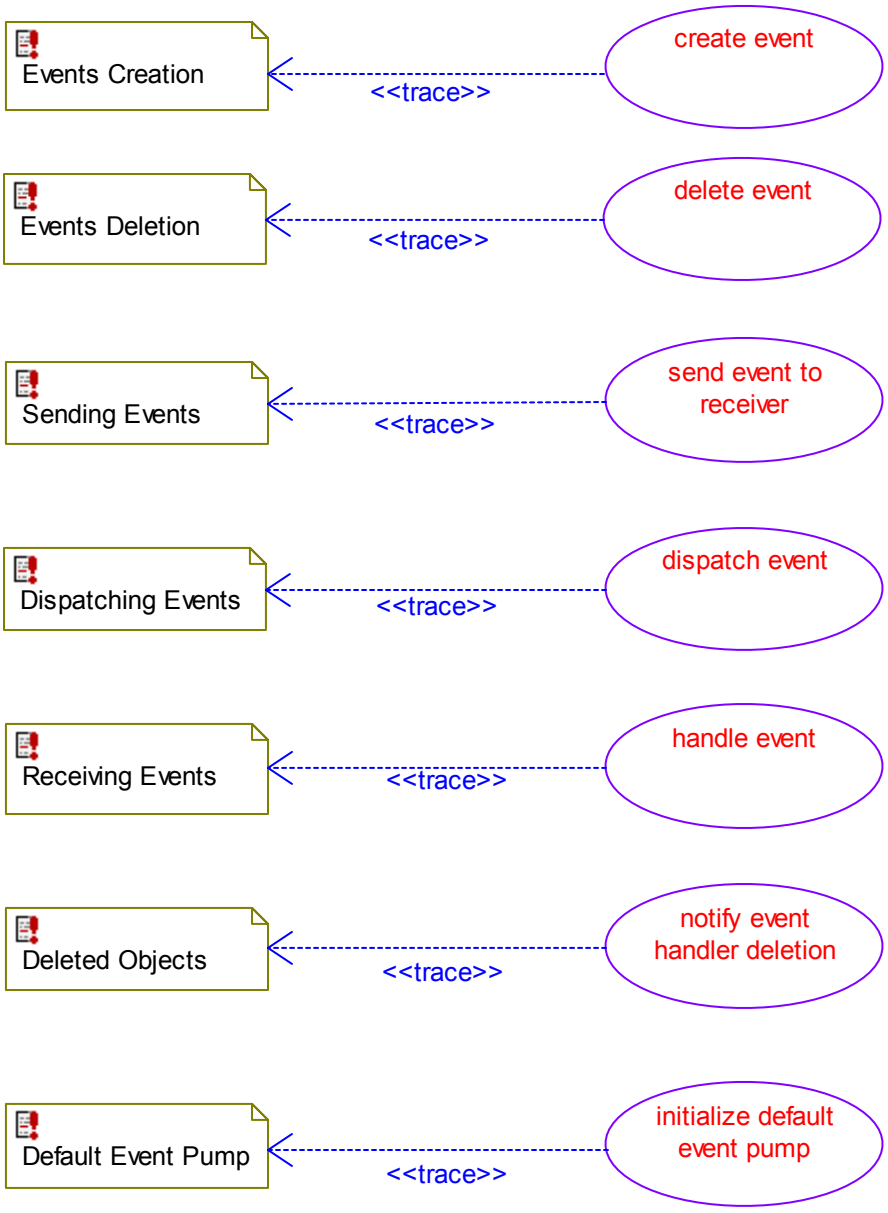
Use Case Diagram name: Handling Events

Description: The event handling use cases



Use Case Diagram name: Requirements Vs Use Cases

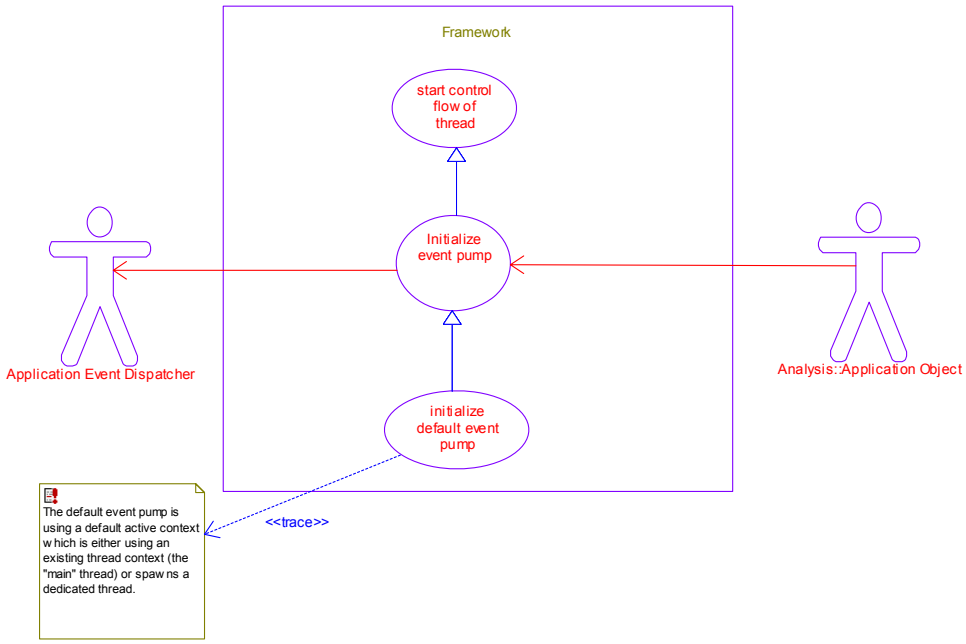
Description: Trace from the domain use cases to the domain requirements.



Use Case Diagram name: Event Handler Deletion
Description: The event handler deletion use cases

Use Case Diagram name: Initializing Event Dispatchers

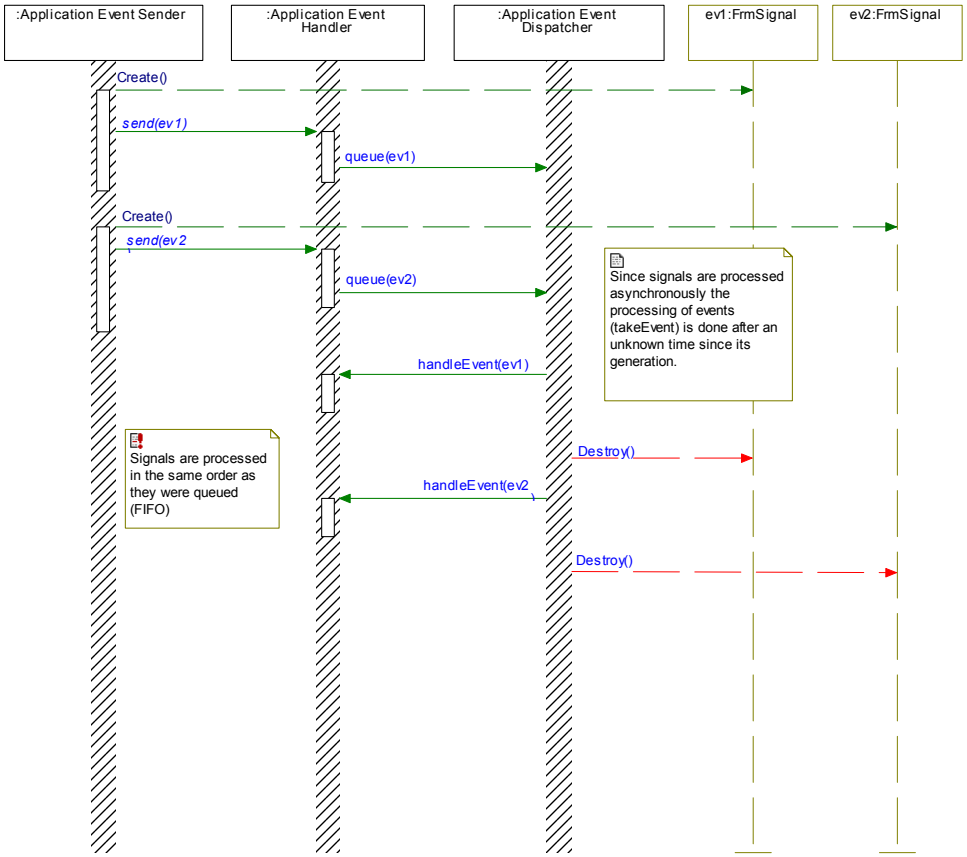
Description: The event dispatcher initialization use cases



Sequence Diagram Information

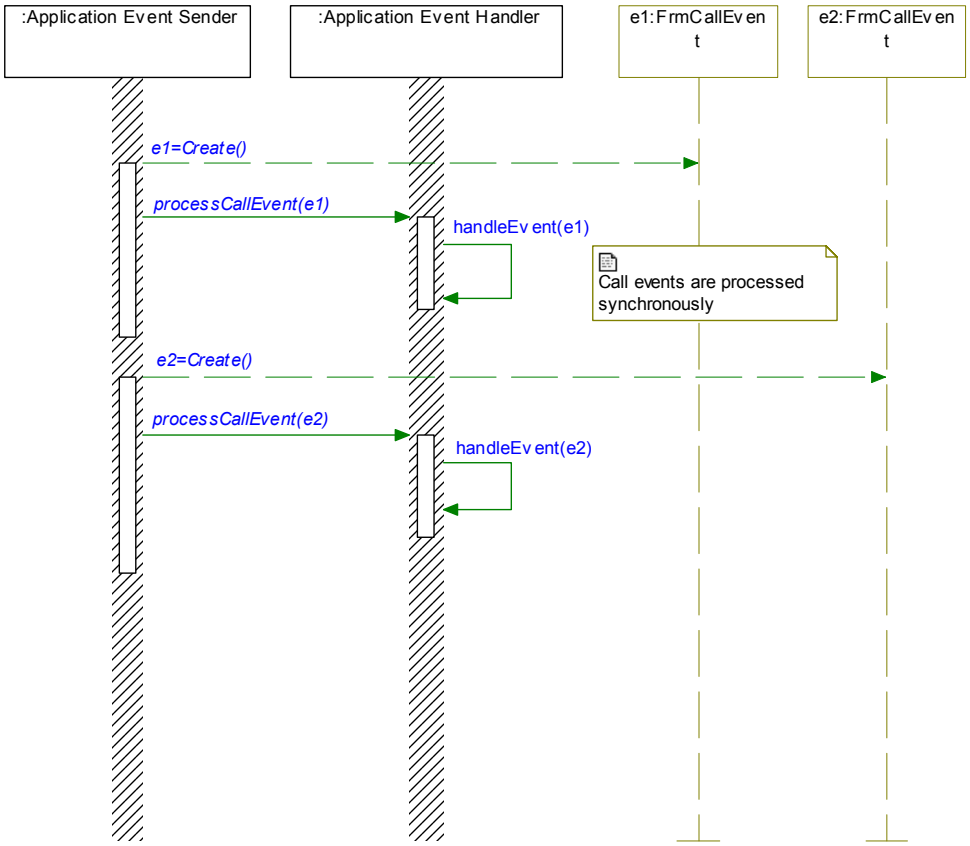
Sequence Diagram name: generate signals and send to receiver

Description: This is the canonical use case of sending a signal to an event receiver. The event is created and then queued.

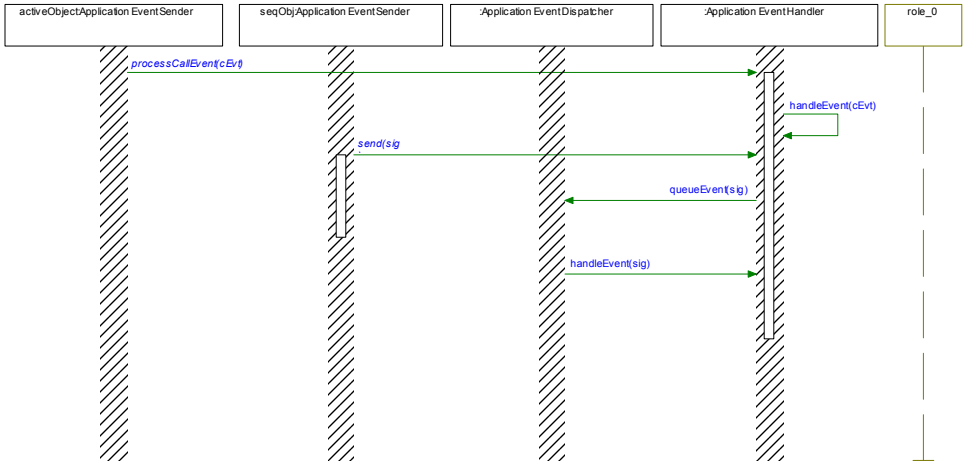


Sequence Diagram name: send call event to receiver

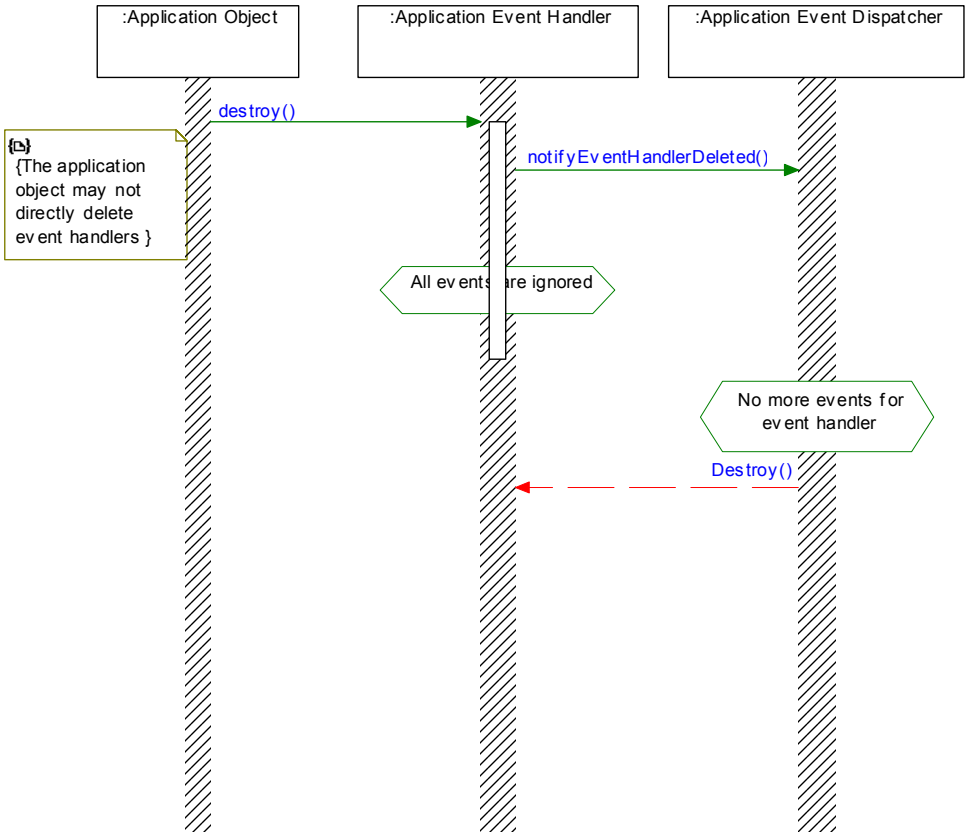
Description: This is the canonical use case of sending a signal to an event receiver. The event is created and then queued.



Sequence Diagram name: a call event is sent and processed and in the meantime a signal arrives
Description: This sequence shows the potential race in event processing.

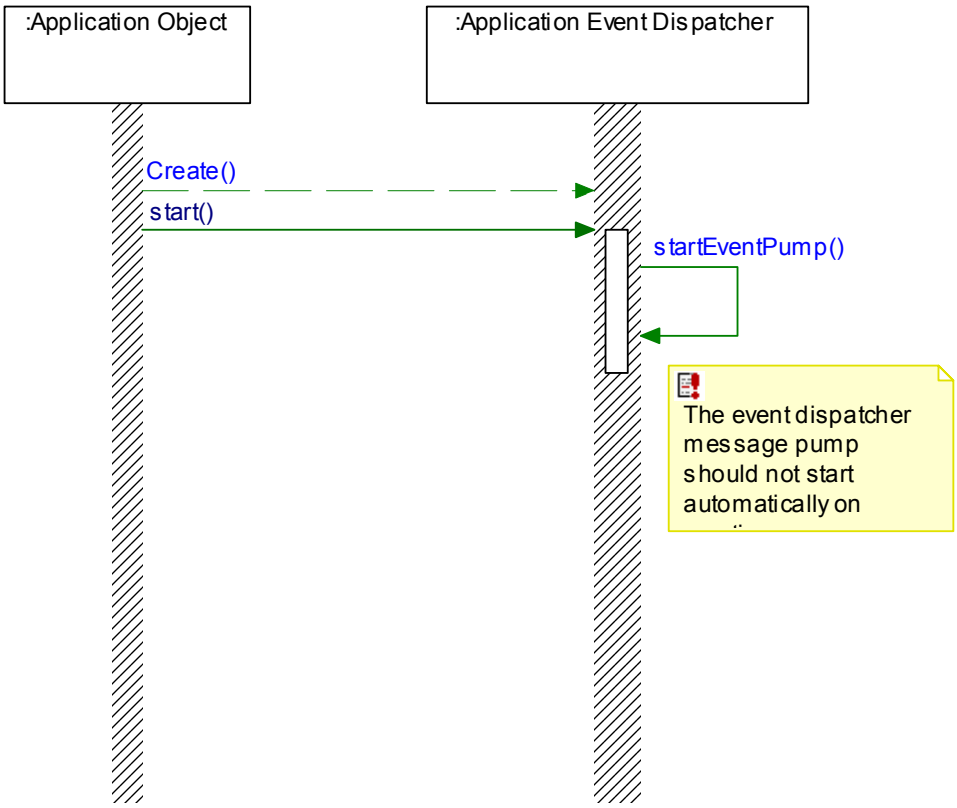


Sequence Diagram name: deletion of event handler
Description: This sequence shows the controlled deletion of an event handler



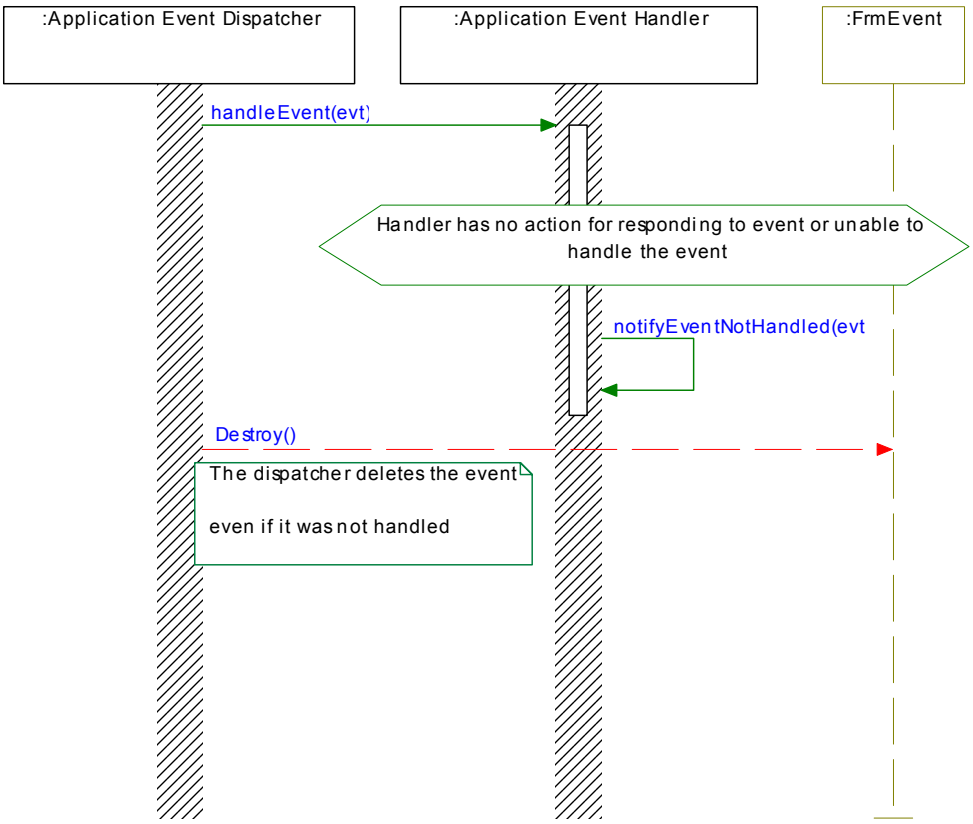
Sequence Diagram name: Initialize Event Pump

Description: This sequence shows the initialization of an event dispatcher



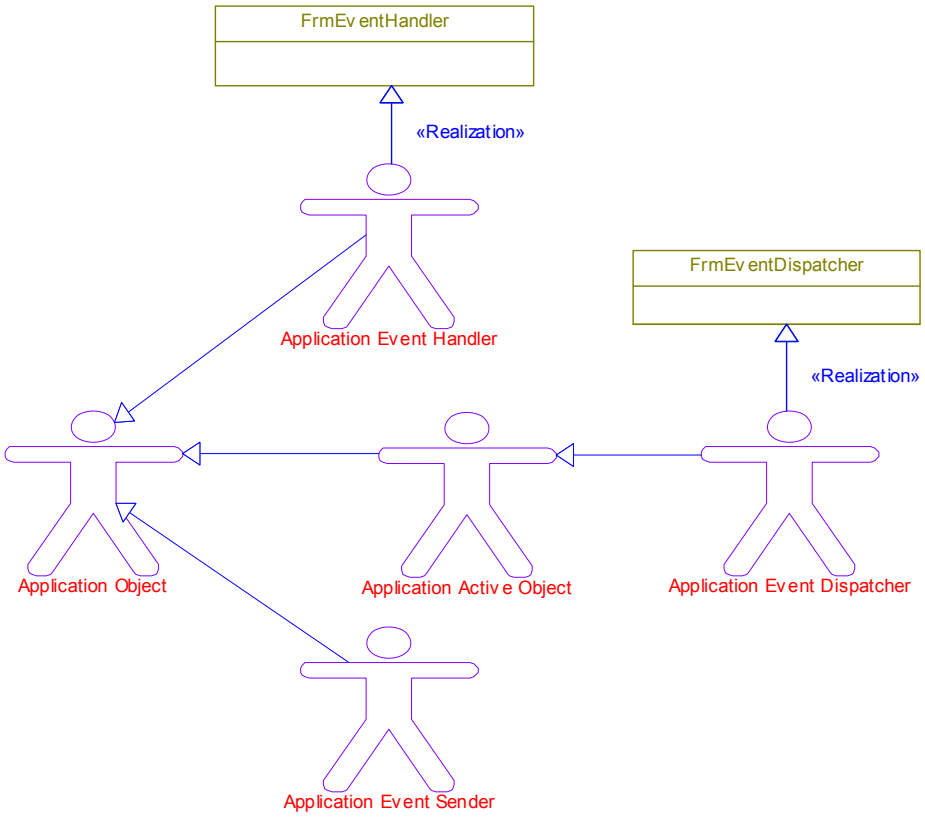
Sequence Diagram name: unhandled events

Description: This sequence shows the reaction to events that cannot be consumed

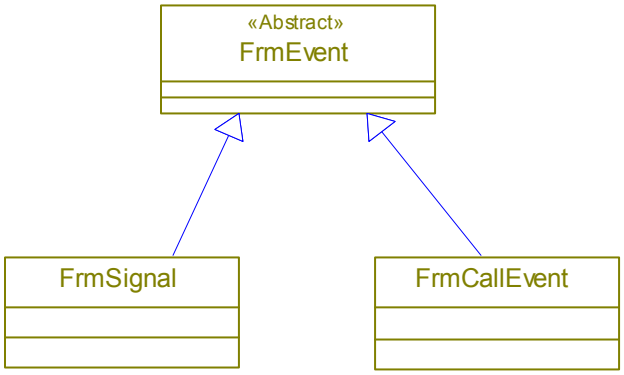


Object Model Diagram Information

Object Model Diagram name: Actors
Description: Actors defined for this domain



Object Model Diagram name: Events
Description: Events defined in this domain



Use Case Information for Package: BasicEventsProcessing

Use Case name: send event to receiver
Description:Send an event to the event handler
Extension Points:

Use Case name: send signal to receiver

Description:Send an asynchronous event to the event handler
Extension Points:

Generalization information for Use Case send signal to receiver

Generalization name: send event to receiver

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
send event to receiver	send event to receiver	send signal to receiver

Use Case name: send call event to receiver

Description:Send a synchronus call-event to the event handler
Extension Points:

Generalization information for Use Case send call event to receiver

Generalization name: send event to receiver

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
send event to receiver	send event to receiver	send call event to receiver

Use Case name: queue event for dispatching

Description:Queue an asynchronous event - to be dispatched later
Extension Points:

Relation information for Use case queue event for dispatching

Relation name: itsApplication Event Dispatcher

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Event Dispatcher
LinkName:
RoleName: itsApplication Event Dispatcher
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Event Dispatcher		queue event for dispatching	Application Event Dispatcher

Use Case name: dispatch event

Description:Dispatch an event to the appropriate handler
Extension Points:

Use Case name: handle signal event

Description:Consume an asynchronous event
Extension Points:

Generalization information for Use Case handle signal event

Generalization name: handle event

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
handle event	handle event	handle signal event

Use Case name: handle call event

Description:Consume a call-event (triggered operation)
Extension Points:

Generalization information for Use Case handle call event

Generalization name: handle event

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
handle event	handle event	handle call event

Use Case name: handle event

Description:Consume an event
Extension Points:

Relation information for Use case handle event

Relation name: itsApplication Event Handler

Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Event Handler
LinkName:
RoleName: itsApplication Event Handler
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Event	itsHandle event	handle event	Application Event

Handler			Handler

Use Case name: notify event was not handled

Description:The event generation succeeded but the event handler did not handle the event - a notification should be sent to the handler
Extension Points:

Relation information for Use case notify event was not handled

Relation name: itsApplication Event Handler

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Event Handler
LinkName:
RoleName: itsApplication Event Handler
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Event Handler		notify event was not handled	Application Event Handler

Use Case name: delete event

Description:Delete an event
Extension Points:

Use Case name: create event

Description:Create an event
Extension Points:

Use Case name: notify event handler deletion

Description:Notify the dispatcher that the event handler is destroyed
Extension Points:

Relation information for Use case notify event handler deletion

Relation name: itsApplication Event Dispatcher

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Event Dispatcher
LinkName:
RoleName: itsApplication Event Dispatcher
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Event Dispatcher		notify event handler deletion	Application Event Dispatcher

Use Case name: Initialize event pump

Description:Initializing the event pump is a special kind of start control flow:
the control flow is an infinite loop waiting for events.
Extension Points:

Generalization information for Use Case Initialize event pump

Generalization name: start control flow of thread

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
start control flow of thread	start control flow of thread	Initialize event pump

Relation information for Use case Initialize event pump

Relation name: itsApplication Event Dispatcher

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Event Dispatcher
LinkName:
RoleName: itsApplication Event Dispatcher
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Event Dispatcher		Initialize event pump	Application Event Dispatcher

Use Case name: initialize default event pump

Description:Start the default event dispatcher
Extension Points:

Generalization information for Use Case initialize default event pump

Generalization name: Initialize event pump

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
Initialize event pump	Initialize event pump	initialize default event pump

Actor Information for Package: BasicEventsProcessing

Actor name: Application Event Handler

Description:An application object that receives the event and processes it (or not).

This actor may use the mechanisms provided in the framework (EventReceiver) however the processing of events is application specific (how the object react to the various events)
In addition, the application class may override some or all of the reception mechanisms of the framework.

In addition, the actual object that receive the events are instances of application classes hence we need an actor to represent the event reception objects.

Generalization information for Actor Application Event Handler

Generalization name: Application Object

Description:
Virtual: false
Visibility: public
Extension Point:

Generalization name: FrmEventHandler

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
Application Object	Application Object	Application Event Handler
FrmEventHandler	FrmEventHandler	Application Event Handler

Relation information for Actor Application Event Handler

Relation name: itsHandle event

Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsHandle event
LinkName:
RoleName: itsHandle event
Type: Association
Description:

Relation name: itsNotify event handler deletion

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsNotify event handler deletion
LinkName:
RoleName: itsNotify event handler deletion
Type: Association
Description:

Name	Inverse	Source	Target
itsHandle event	itsApplication Event Handler	Application Event Handler	handle event
itsNotify event handler deletion		Application Event Handler	notify event handler deletion

Actor name: Application Event Sender

Description:An application object that sends an event to another (receiver) object
Any application object can play the role of event sender

Generalization information for Actor Application Event Sender

Generalization name: Application Object

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
Application Object	Application Object	Application Event Sender

Relation information for Actor Application Event Sender

Relation name: itsSend event to receiver

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsSend event to receiver
LinkName:
RoleName: itsSend event to receiver
Type: Association
Description:

Relation name: itsSend start statechart event

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsSend start statechart event
LinkName:
RoleName: itsSend start statechart event
Type: Association
Description:

Relation name: itsTerminate statechart

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsTerminate statechart
LinkName:
RoleName: itsTerminate statechart
Type: Association
Description:

Name	Inverse	Source	Target
itsSend event to		Application Event	send event to receiver

receiver		Sender	
itsSend start statechart event		Application Event Sender	send start statechart event
itsTerminate statechart		Application Event Sender	terminate statechart

Actor name: Application Event Dispatcher

Description:Dispatches events from a queue to event receivers.

The framework has a complete dispatching mechanism (EventDispatcher class) and the application is not required to implement any part of the dispatching mechanisms.

Nevertheless, the application must instantiate event dispatchers, and these are usually instances of classes that inherit the framework event dispatcher. These classes may or may not override the functionality of the framework event dispatcher.

Since the application object (even if this is the default\main thread) serve as dispatchers we specify an actor that extends the framework EventDIspatcher analysis class.

The extension can be by inheritance or delegation

Generalization information for Actor Application Event Dispatcher

Generalization name: Application Active Object

Description:
Virtual: false
Visibility: public
Extension Point:

Generalization name: FrmEventDispatcher

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
Application Active Object	Application Active Object	Application Event Dispatcher
FrmEventDispatcher	FrmEventDispatcher	Application Event Dispatcher

Relation information for Actor Application Event Dispatcher

Relation name: itsDispatch event

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsDispatch event
LinkName:
RoleName: itsDispatch event
Type: Association
Description:

Name	Inverse	Source	Target
itsDispatch event		Application Event	dispatch event

		Dispatcher	

Class Information for Package: BasicEventsProcessing

Class name: FrmEventDispatcher

Description:Implements the dispatching mechanisms of events from the event pool to their destinations.

The framework has a complete dispatching mechanism and the application is not required to implement parts of the dispatching mechanisms.
Nevertheless, the application must instantiate event dispatchers, and these are usually instances of classes that inherit the framework event dispatcher. These class may or may not override the functionality of the framework event dispatcher.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Class name: FrmEventHandler

Description:Implements the generic mechanism to receive and process events.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Class name: FrmEvent

Description:An abstract class signifying an occurrence that may trigger behavior
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Class name: FrmSignal

Description:An asynchronous event
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Generalization information for Class FrmSignal

Generalization name: FrmEvent

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
FrmEvent	FrmEvent	FrmSignal

Class name: FrmCallEvent

Description:A synchronous event, sometimes also referred to as a triggered operation
Active: false

Behavior Overridden: false
Composite: false
Reactive: false

Generalization information for Class FrmCallEvent

Generalization name: FrmEvent

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
FrmEvent	FrmEvent	FrmCallEvent

Constraint information for Package BasicEventsProcessing

Constraint name: deleteEventHandler

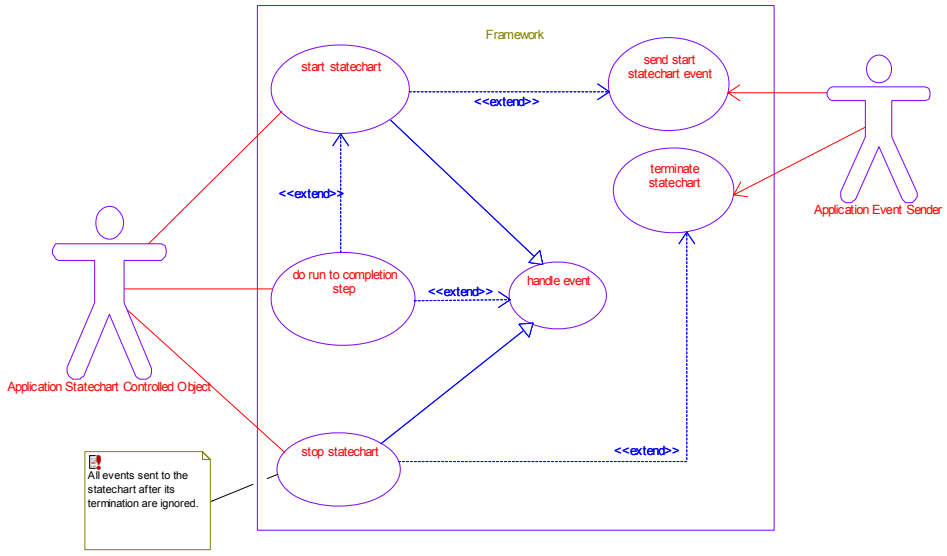
Body: The application object may not directly delete event handlers
Description:

Package: StatechartManagement

Use Case Diagram Information

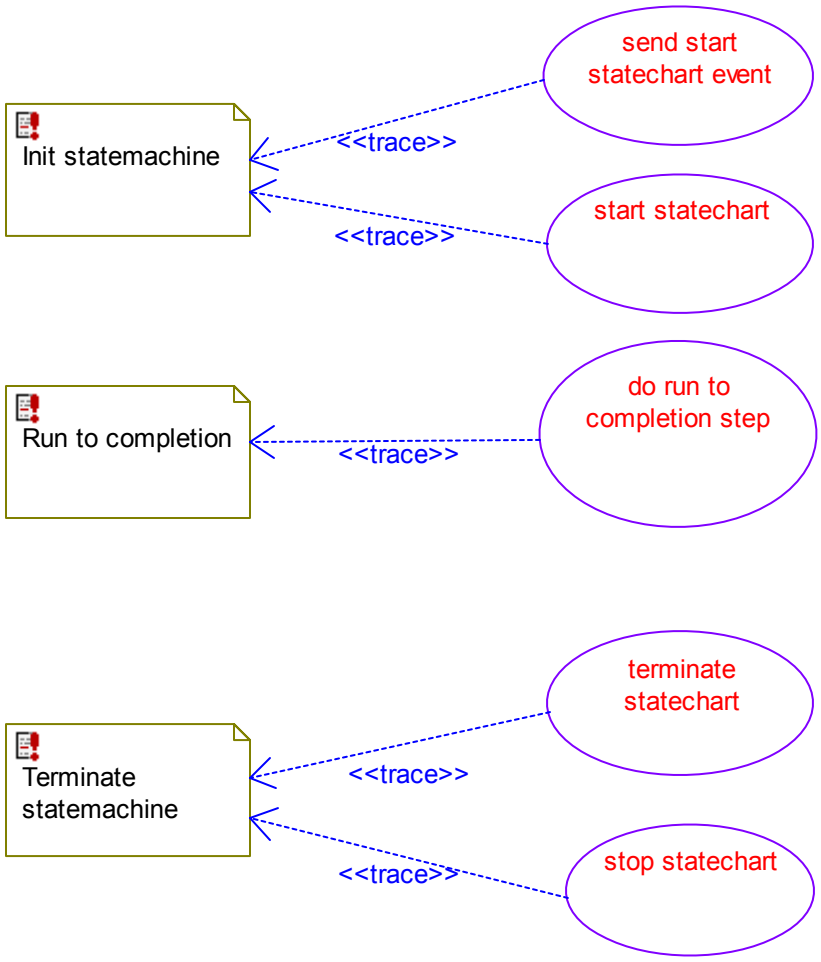
Use Case Diagram name: Overview

Description: The domain use cases



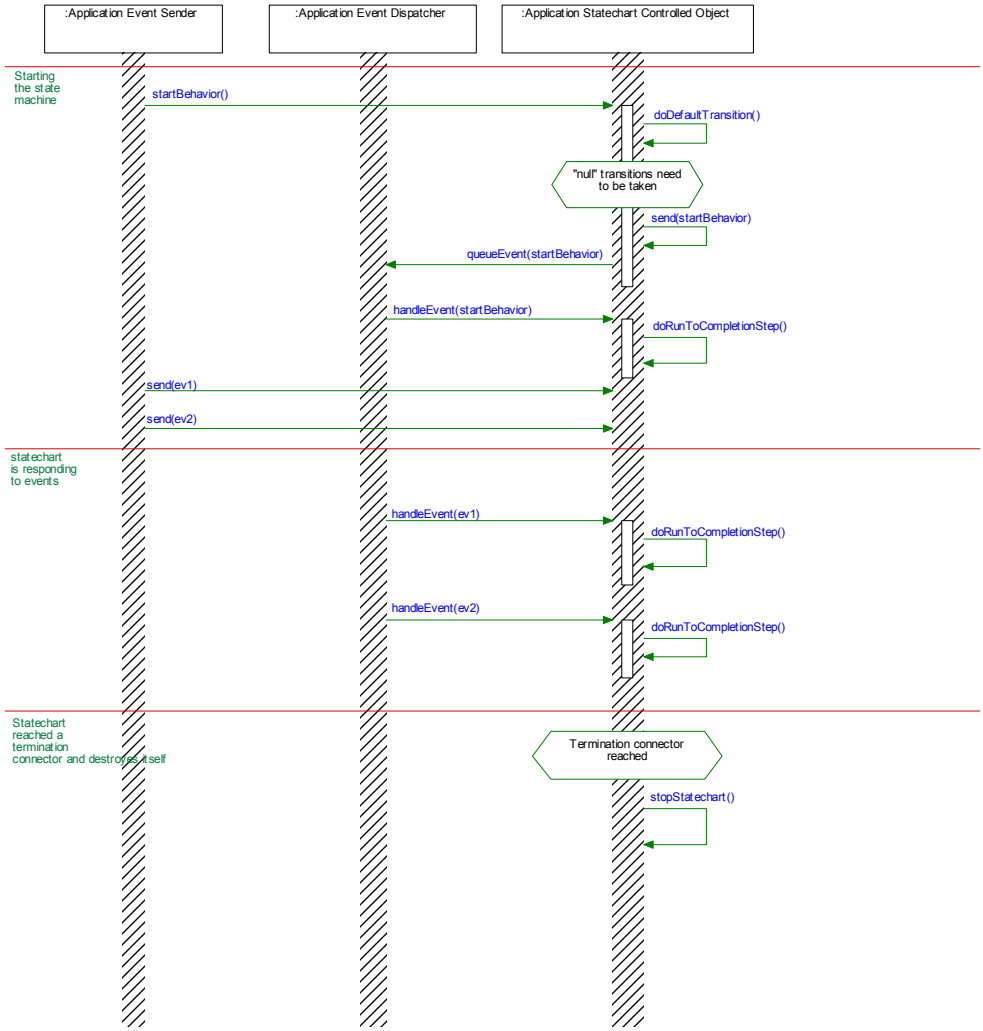
Use Case Diagram name: Requirements Vs Usecases

Description: Trace from the domain use cases to the domain requirements.



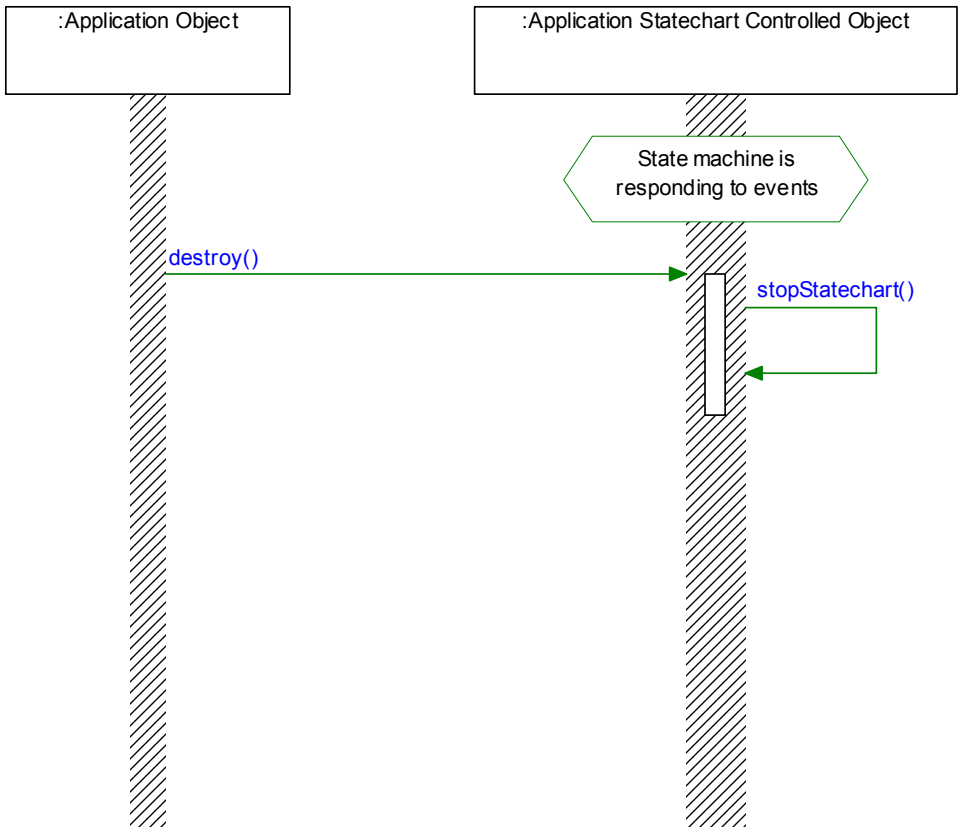
Sequence Diagram Information

Sequence Diagram name: statemachine life cycle scenario
Description: This sequence shows the life cycle of an event handler



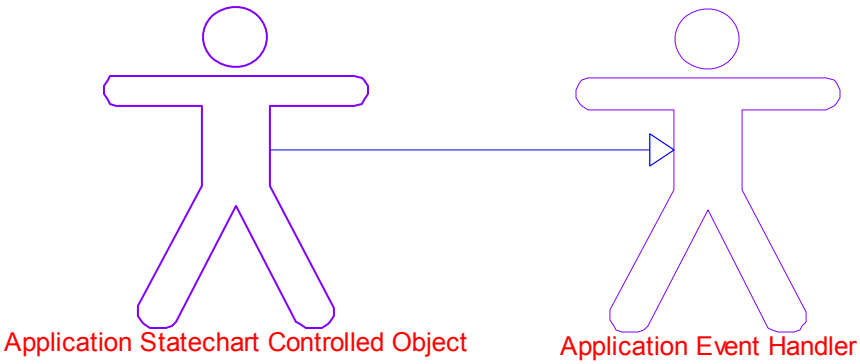
Sequence Diagram name: external termination

Description: This sequence shows the termination of an event handler by another object



Object Model Diagram Information

Object Model Diagram name: Actors
Description: Actors defined for this domain



Use Case Information for Package: StatechartManagement

Use Case name: start statechart

Description:Start the state machine
Extension Points:

Generalization information for Use Case start statechart

Generalization name: handle event

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
handle event	handle event	start statechart

Relation information for Use case start statechart

Relation name: itsApplication Statechart Controlled Object

Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Statechart Controlled Object
LinkName:
RoleName: itsApplication Statechart Controlled Object
Type: Association
Description:

Name	Inverse	Source	Target

itsApplication Statechart Controlled Object	itsStart statechart	start statechart	Application Statechart Controlled Object

Use Case name: stop statechart

Description:Stop the state machine
Extension Points:

Generalization information for Use Case stop statechart

Generalization name: handle event

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
handle event	handle event	stop statechart

Relation information for Use case stop statechart

Relation name: itsApplication Statechart Controlled Object

Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Statechart Controlled Object
LinkName:
RoleName: itsApplication Statechart Controlled Object
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Statechart Controlled Object	itsStop statechart	stop statechart	Application Statechart Controlled Object

Use Case name: send start statechart event

Description:Send a start event to the state machine
Extension Points:

Use Case name: terminate statechart

Description:Destroy the state machine
Extension Points:

Use Case name: do run to completion step

Description:if the current state has a transition triggered by the event, the Application Statechart Controller does the run-to-completion step as specified in the statechart.
Extension Points:

Relation information for Use case do run to completion step

Relation name: itsApplication Statechart Controlled Object

Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Statechart Controlled Object
LinkName:
RoleName: itsApplication Statechart Controlled Object
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Statechart Controlled Object	itsDo run to completion step	do run to completion step	Application Statechart Controlled Object

Actor Information for Package: StatechartManagement

Actor name: Application Statechart Controlled Object

Description:An application statechart controlled object is an application object whose (part of its) behavior is specified by a statechart.
These objects react to events that may trigger transitions of statecharts.

Generalization information for Actor Application Statechart Controlled Object

Generalization name: Application Event Handler

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
Application Event Handler	Application Event Handler	Application Statechart Controlled Object

Relation information for Actor Application Statechart Controlled Object

Relation name: itsStart statechart

Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsStart statechart
LinkName:
RoleName: itsStart statechart
Type: Association
Description:

Relation name: itsStop statechart

Symmetric: true

Multiplicity: 1
Qualifier:
Visibility: public
Label: itsStop statechart
LinkName:
RoleName: itsStop statechart
Type: Association
Description:

Relation name: itsDo run to completion step
Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsDo run to completion step
LinkName:
RoleName: itsDo run to completion step
Type: Association
Description:

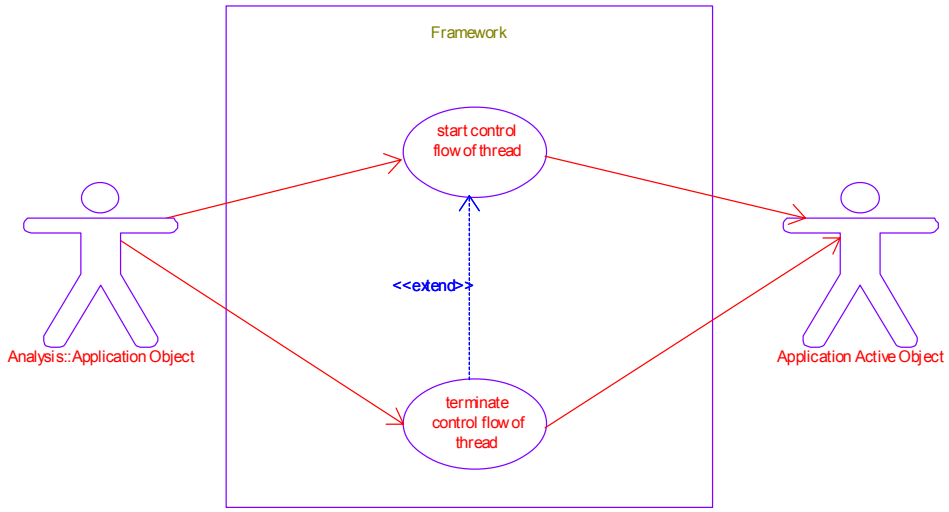
Name	Inverse	Source	Target
itsStart statechart	itsApplication Statechart Controlled Object	Application Statechart Controlled Object	start statechart
itsStop statechart	itsApplication Statechart Controlled Object	Application Statechart Controlled Object	stop statechart
itsDo run to completion step	itsApplication Statechart Controlled Object	Application Statechart Controlled Object	do run to completion step

Package: ThreadsManagement

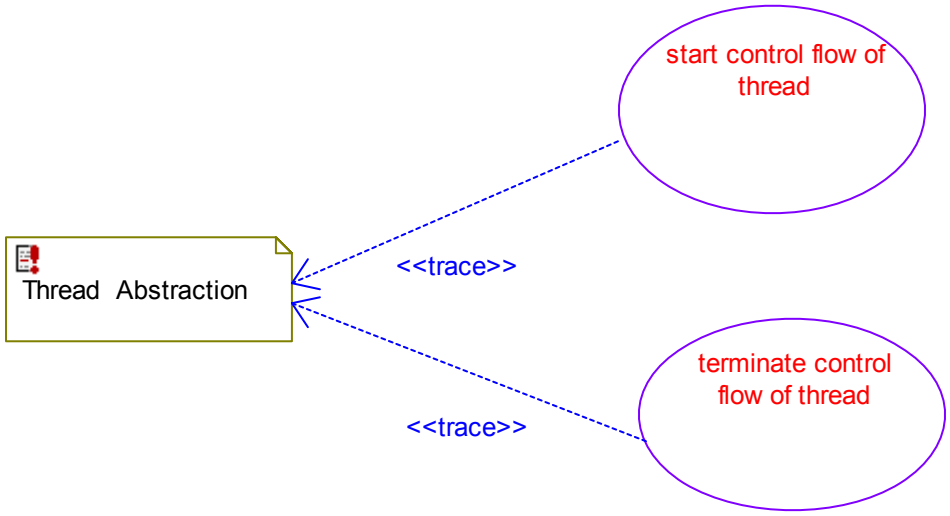
Use Case Diagram Information

Use Case Diagram name: Threads abstractions

Description: The domain use cases

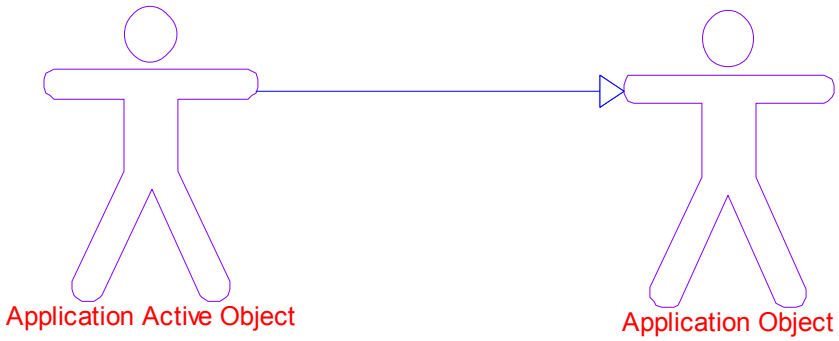


Use Case Diagram name: Requirements Vs Use Cases
Description: Trace from the domain use cases to the domain requirements.



Object Model Diagram Information

Object Model Diagram name: Actors
Description: Actors defined for this domain



Use Case Information for Package: ThreadsManagement

Use Case name: start control flow of thread
Description:Control over the thread execution and state
Extension Points:

Relation information for Use case start control flow of thread

Relation name: itsApplication Active Object
Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Active Object
LinkName:
RoleName: itsApplication Active Object

Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Active Object		start control flow of thread	Application Active Object

Use Case name: terminate control flow of thread

Description:Destroy a controlled thread
Extension Points:

Relation information for Use case terminate control flow of thread

Relation name: itsApplication Active Object

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Active Object
LinkName:
RoleName: itsApplication Active Object
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Active Object		terminate control flow of thread	Application Active Object

Actor Information for Package: ThreadsManagement

Actor name: Application Active Object

Description:An Application Active Object is an object that controls its thread of execution

Generalization information for Actor Application Active Object

Generalization name: Application Object

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
Application Object	Application Object	Application Active Object

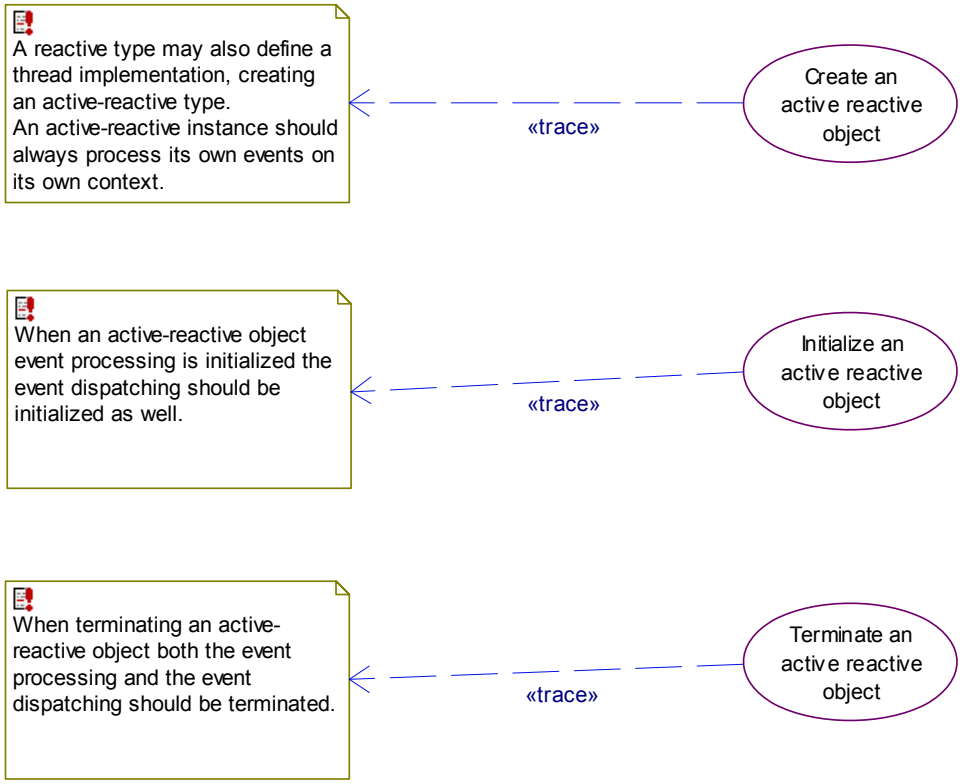
Package Information

Description: This package contains the requirement and use-case analysis of the certifiable framework thread management domain.

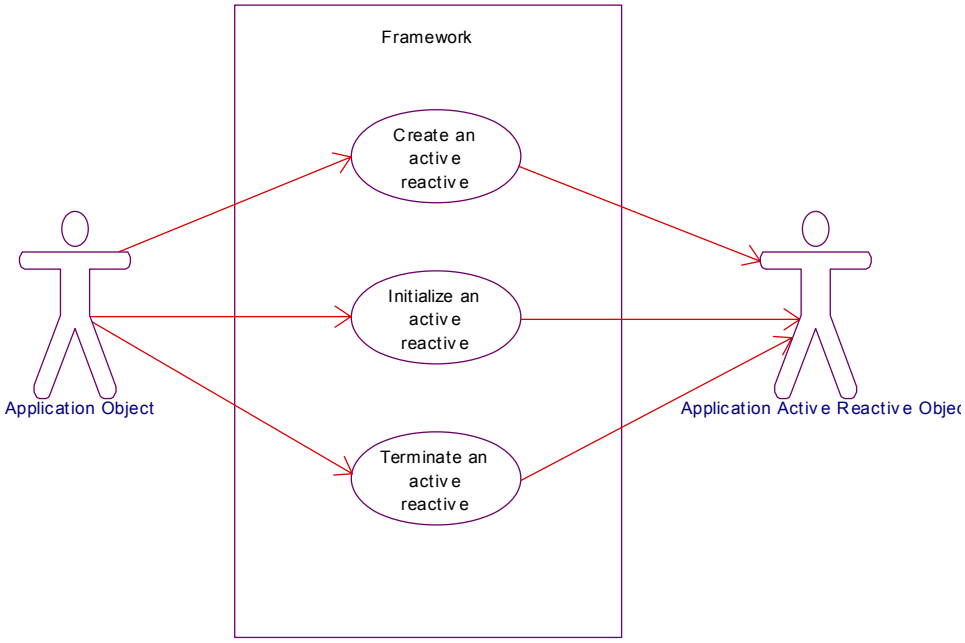
Package: ReactiveThreads

Use Case Diagram Information

Use Case Diagram name: Requirements Vs Use Cases
Description: Trace from the domain use cases to the domain requirements.

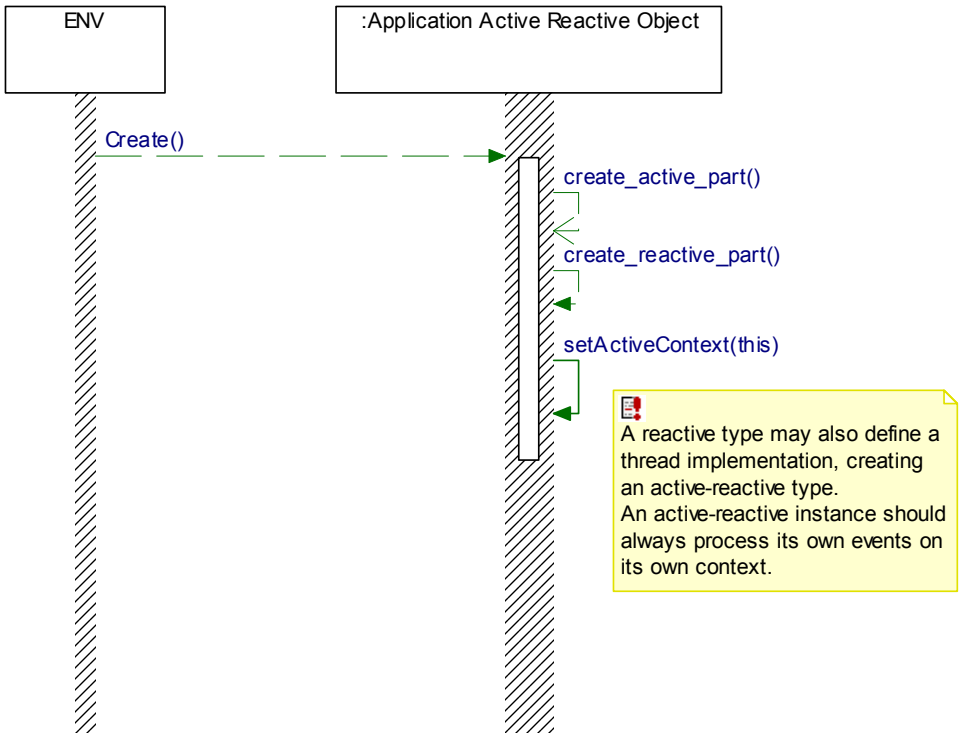


Use Case Diagram name: Reactive threads
Description: The domain use cases

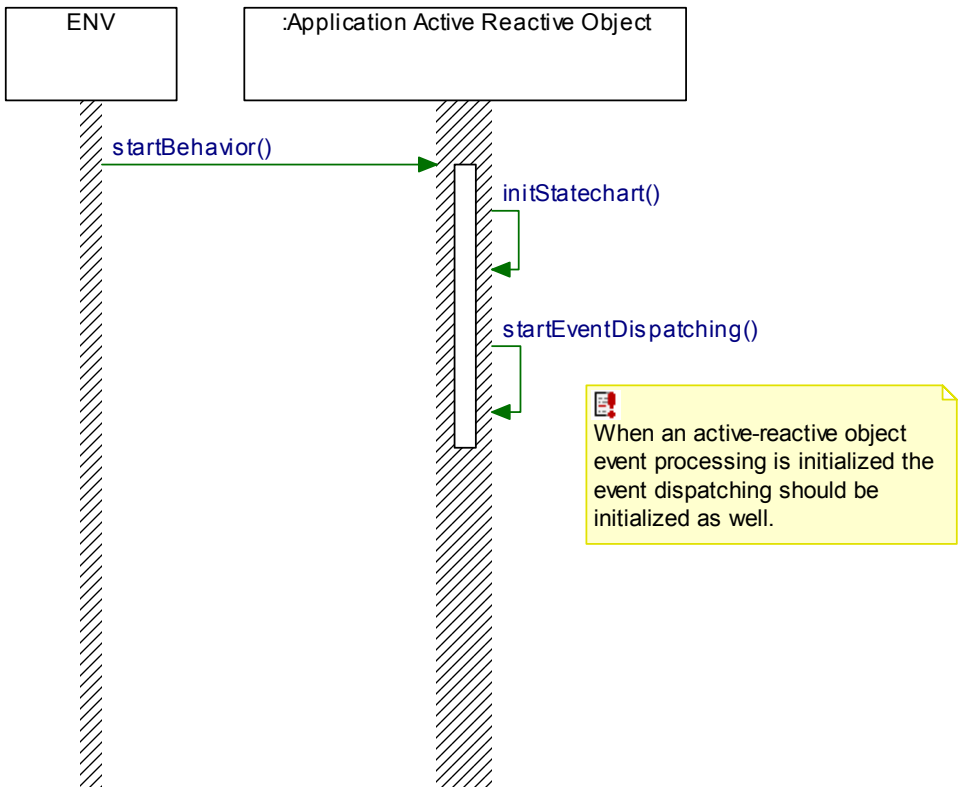


Sequence Diagram Information

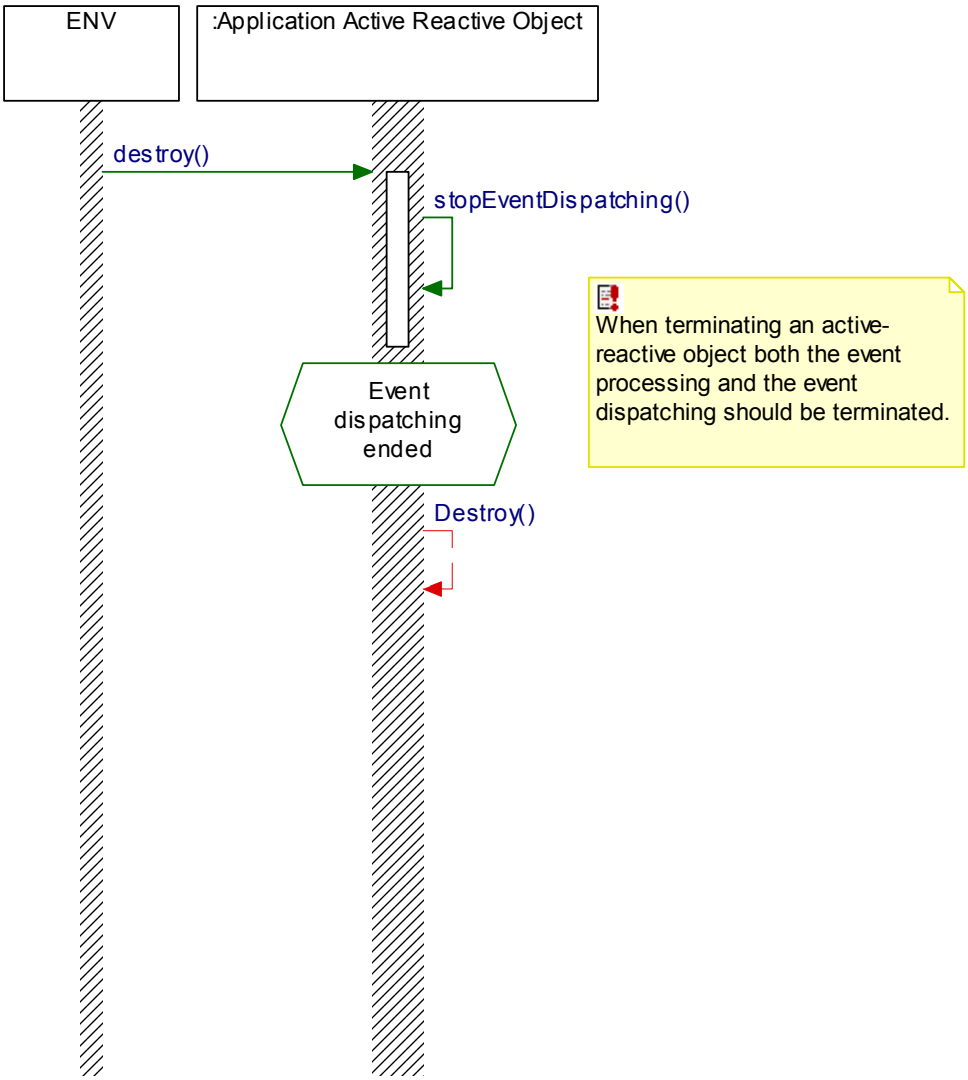
Sequence Diagram name: Active reactive instance creation
Description: This sequence shows the creation of an active-reactive instance.



Sequence Diagram name: Initialize active reactive behavior
Description: This sequence shows the initialization of an active-reactive instance behavior.



Sequence Diagram name: Terminate active reactive behavior
Description: This sequence shows the termination of an active-reactive instance behavior.



Use Case Information for Package: ReactiveThreads

Use Case name: Create an active reactive object

Description:Create an active-reactive instance
Extension Points:

Relation information for Use case Create an active reactive object

Relation name: itsApplication Active Reactive Object

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Active Reactive Object
LinkName:
RoleName: itsApplication Active Reactive Object
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Active Reactive Object		Create an active reactive object	Application Active Reactive Object

Use Case name: Initialize an active reactive object

Description:Initialize the active-reactive instance behavior
Extension Points:

Relation information for Use case Initialize an active reactive object

Relation name: itsApplication Active Reactive Object

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Active Reactive Object
LinkName:
RoleName: itsApplication Active Reactive Object
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Active Reactive Object		Initialize an active reactive object	Application Active Reactive Object

Use Case name: Terminate an active reactive object

Description:terminate the active reactive instance behavior
Extension Points:

Relation information for Use case Terminate an active reactive object

Relation name: itsApplication Active Reactive Object

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Active Reactive Object
LinkName:
RoleName: itsApplication Active Reactive Object
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Active Reactive Object		Terminate an active reactive object	Application Active Reactive Object

Actor Information for Package: ReactiveThreads

Actor name: Application Active Reactive Object

Description:An Application Active Object is an object that controls its thread of execution

Generalization information for Actor Application Active Reactive Object

Generalization name: Application Object

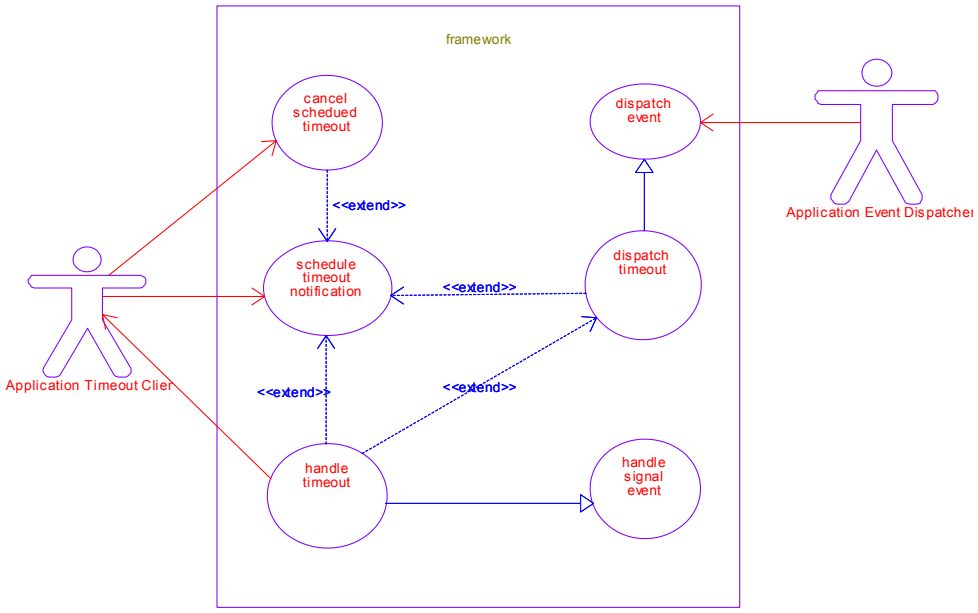
Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
Application Object	Application Object	Application Active Reactive Object

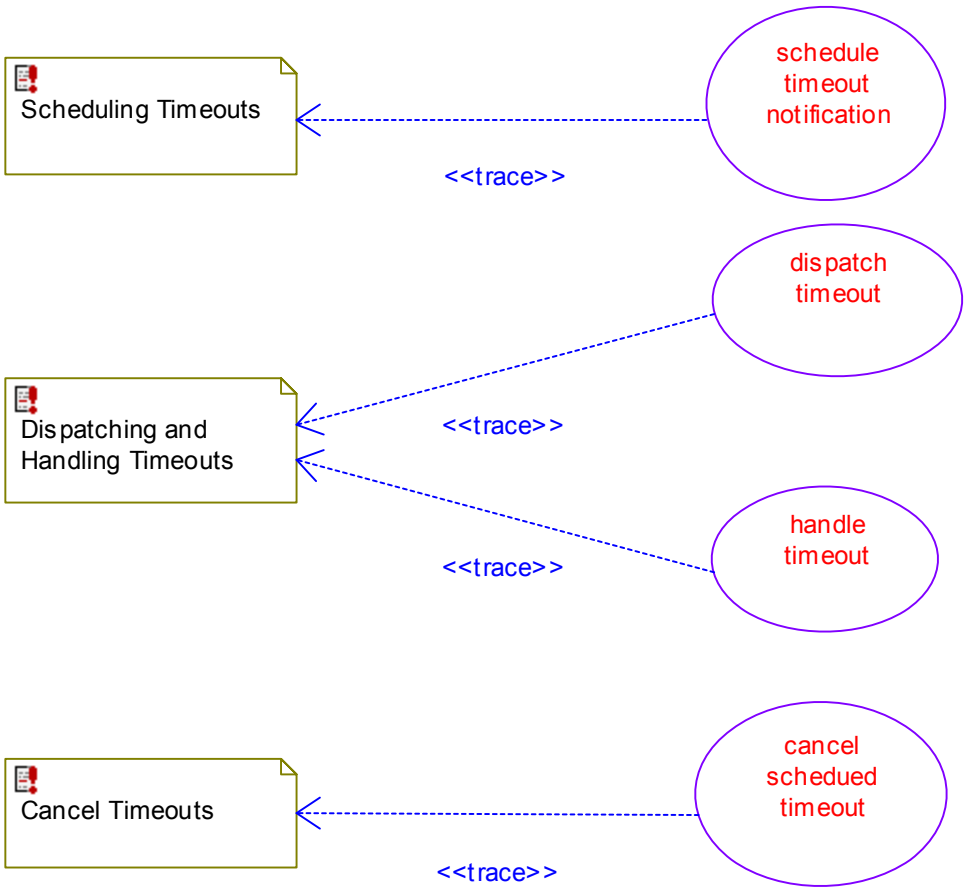
Package: TimeoutManagement

Use Case Diagram Information

Use Case Diagram name: Time Management
Description: The domain use cases

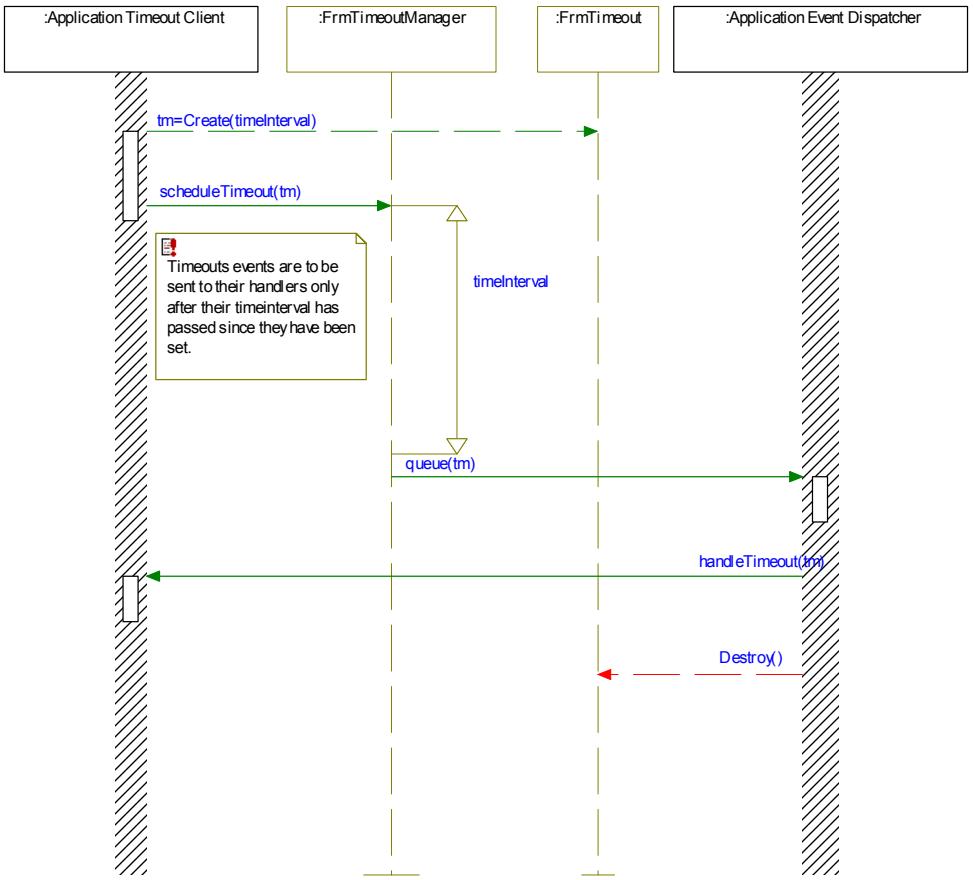


Use Case Diagram name: Requirements Vs UseCases
Description: Trace from the domain use cases to the domain requirements.



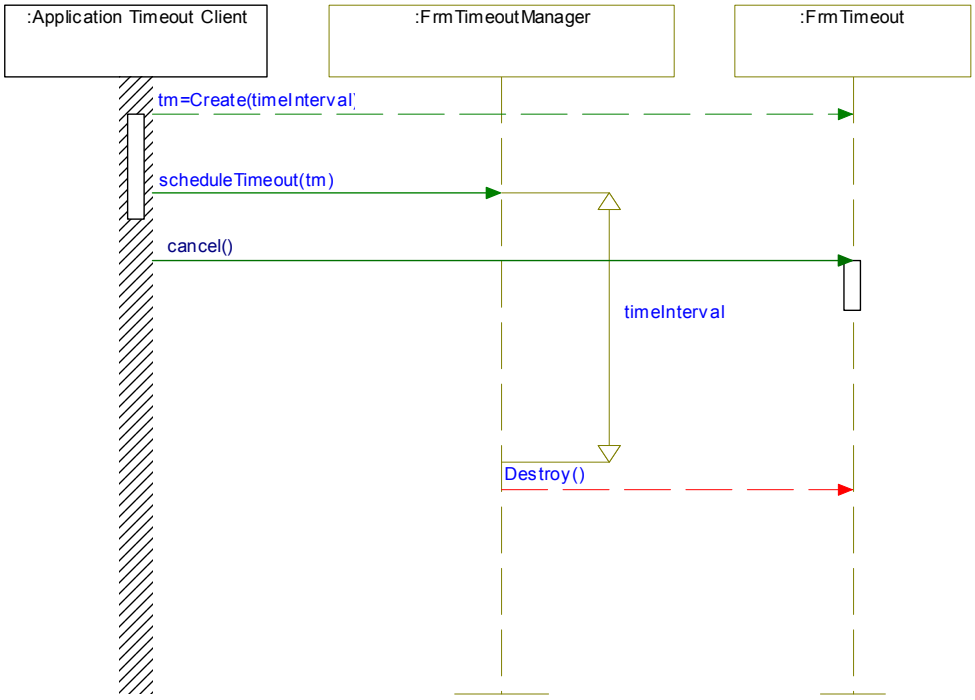
Sequence Diagram Information

Sequence Diagram name: scheduling dispatching and handling of a timeout
Description: This sequence shows the dispatching of a matured timeout



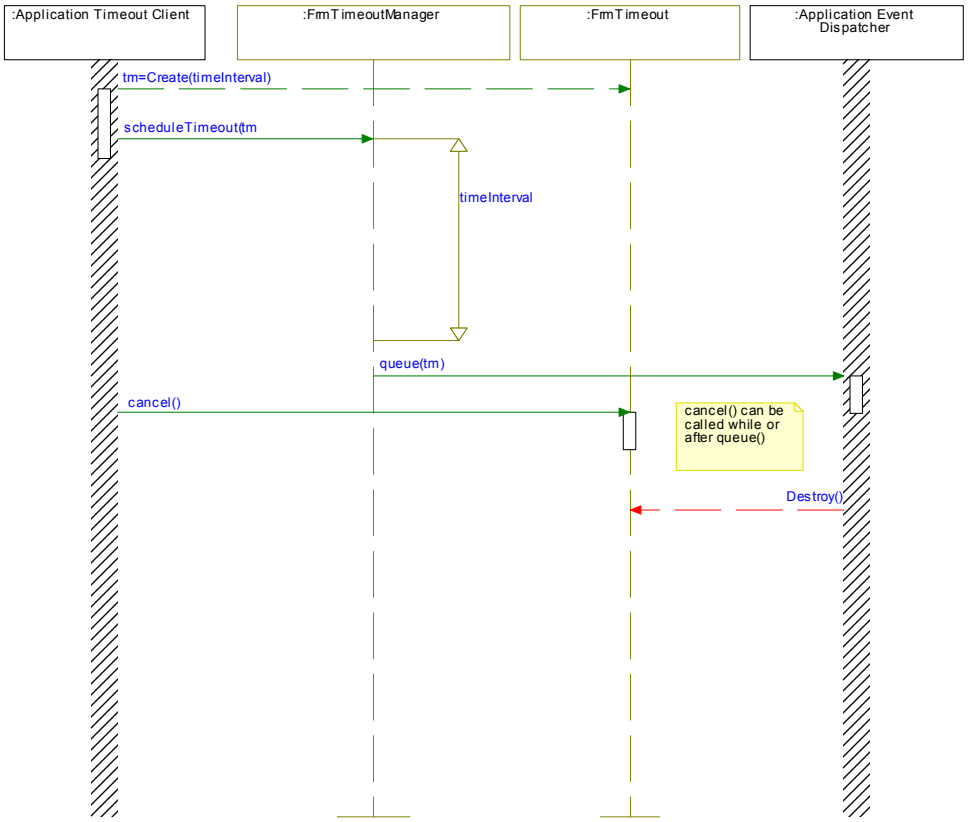
Sequence Diagram name: cancel a scheduled timeout before tm interval passed

Description: This sequence shows a canceling of a pending timeout



Sequence Diagram name: cancel a scheduled timeout after tm interval passed

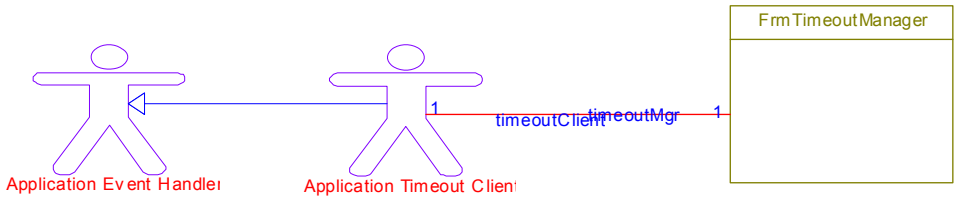
Description: This sequence shows a canceling of a matured timeout



Object Model Diagram Information

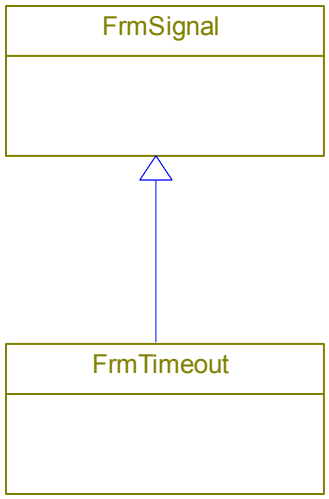
Object Model Diagram name: Actors

Description: Actors defined for this domain



Object Model Diagram name: TimeoutEvents

Description: Events defined in this domain



Use Case Information for Package: TimeoutManagement

Use Case name: handle timeout

Description:Consume a timeout
Extension Points:

Generalization information for Use Case handle timeout

Generalization name: handle signal event

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
handle signal event	handle signal event	handle timeout

Relation information for Use case handle timeout

Relation name: itsApplication Timeout Client

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsApplication Timeout Client
LinkName:
RoleName: itsApplication Timeout Client
Type: Association
Description:

Name	Inverse	Source	Target
itsApplication Timeout Client		handle timeout	Application Timeout Client

Use Case name: send expired timeout event to client

Description:Send a matured timeout to its event handler client
Extension Points:

Use Case name: schedule timeout notification

Description:Schedule a timeout
Extension Points:

Use Case name: cancel schedued timeout

Description:Canceling a scheduled timeout
Extension Points:

Use Case name: dispatch timeout

Description:After the timeout duration passes the timeout event is dispatched
Extension Points:

Generalization information for Use Case dispatch timeout

Generalization name: dispatch event

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
dispatch event	dispatch event	dispatch timeout

Actor Information for Package: TimeoutManagement

Actor name: Application Timeout Client

Description:The Application Timeout Client is an Application Event Handler that requests to receive notifications (timeout events) from the framework, using the FrmTimeoutManager

Generalization information for Actor Application Timeout Client

Generalization name: Application Event Handler

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
Application Event Handler	Application Event Handler	Application Timeout Client

Relation information for Actor Application Timeout Client

Relation name: timeoutMgr

Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public

Label: timeoutMgr
LinkName:
RoleName: timeoutMgr
Type: Association
Description:

Relation name: itsSchedule timeout notification

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsSchedule timeout notification
LinkName:
RoleName: itsSchedule timeout notification
Type: Association
Description:

Relation name: itsCancel schedued timeout

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsCancel schedued timeout
LinkName:
RoleName: itsCancel schedued timeout
Type: Association
Description:

Name	Inverse	Source	Target
timeoutMgr	timeoutClient	Application Timeout Client	FrmTimeoutManager
itsSchedule timeout notification		Application Timeout Client	schedule timeout notification
itsCancel schedued timeout		Application Timeout Client	cancel schedued timeout

Class Information for Package: TimeoutManagement

Class name: FrmTimeoutManager

Description:The framework timeout manager
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Relation information for Class FrmTimeoutManager

Relation name: timeoutClient

Symmetric: true
Multiplicity: 1
Qualifier:
Visibility: public
Label: timeoutClient
LinkName:

RoleName: timeoutClient
Type: Association
Description:

Name	Inverse	Source	Target
timeoutClient	timeoutMgr	FrmTimeoutManager	Application Timeout Client

Class name: FrmTimeout

Description:A timeout event. Note that these are always asynchronous.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Generalization information for Class FrmTimeout

Generalization name: FrmSignal
Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
FrmSignal	FrmSignal	FrmTimeout

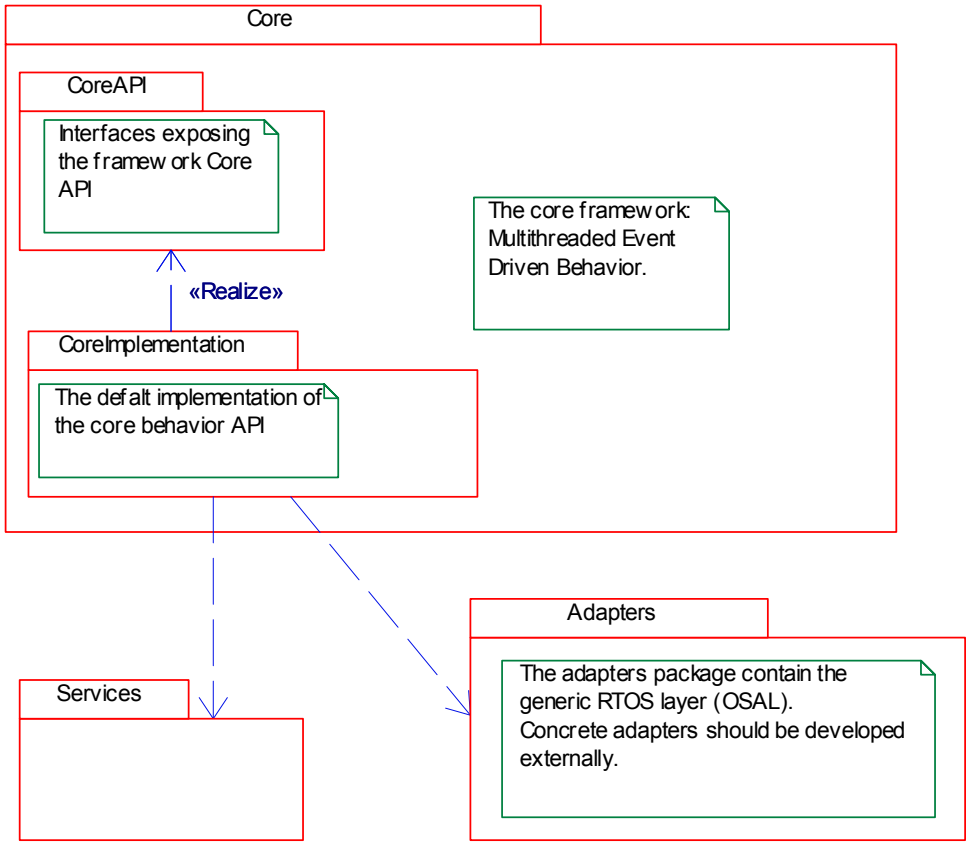
Package: Design

Description: The Object eXecution Framework package

Object Model Diagram Information

Object Model Diagram name: Packages overview

Description: This diagram shows the various packages within the OXF and the OXF API interfaces.



Package Information

Description: The Object eXecution Framework package

Package: aom

Class Information for Package: aom

Class name: AOMSState

Description:Instrumentation state vector data class
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Class name: AOMInstance

Description:An application instance reflection
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Class name: AnimServices

Description:The AOM entry point for the OXF
Active: false

Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: AnimServices

Operation name: notifyEndApplication
Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyEndApplication(bool closeConnection)
Return Type: void
Description: notify the AOM that the application is about to terminate.
The AOM should close the connection to Rhapsody.

Argument information for Operation notifyEndApplication

Name	Type	Direction
closeConnection	bool	In

Operation name: isApplicationEnding
Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: isApplicationEnding()
Return Type: bool
Description: Query if the application is terminating, to be used by various AOM algorithms instead of query the OXF implementation

Operation name: registerAnimTimerManager
Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: registerAnimTimerManager(IOxfAnimTimerManager timerManager)
Return Type: void
Description: Register the IOxfAnimTimerManager on the AOM to enable time control (stop timeouts in breakpoints, advance the time on go idle commands, etc.)

Argument information for Operation registerAnimTimerManager

Name	Type	Direction
timerManager	IOxfAnimTimerManager	In

Operation name: notifyMutexLock

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyMutexLock(bool locked)
Return Type: void
Description: Notify the animation that a thread is going through a mutex.
The operation should be called twice, before and after the lock.

Argument information for Operation notifyMutexLock

Name	Type	Direction
locked	bool	In

Operation name: notifyEventDestroyed

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyEventDestroyed(IOxfEvent ev)
Return Type: void
Description: Notify that an event is about to be destroyed

Argument information for Operation notifyEventDestroyed

Name	Type	Direction
ev	IOxfEvent	In

Operation name: notifyStartBehaviorBegin

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyStartBehaviorBegin(IOxfReactive instance)
Return Type: void
Description: Notify that the behavior (i.e. statemachine) of a reactive instance is about to be activated.

Argument information for Operation notifyStartBehaviorBegin

Name	Type	Direction
instance	IOxfReactive	In

Operation name: notifyHandleEventBegin

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyHandleEventBegin(IOxfReactive instance,IOxfEvent ev)
Return Type: void
Description: Notify that an event is about to be handled

Argument information for Operation notifyHandleEventBegin

Name	Type	Direction
instance	IOxfReactive	In
ev	IOxfEvent	In

Operation name: notifyNullTransition

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyNullTransition(IOxfReactive instance)
Return Type: void
Description: Notify that a null-transition is about to be taken.

Argument information for Operation notifyNullTransition

Name	Type	Direction
instance	IOxfReactive	In

Operation name: notifyStateEntered

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyStateEntered(IOxfReactive instance,char* state)
Return Type: void
Description: Notify that a state was entered

Argument information for Operation notifyStateEntered

Name	Type	Direction
instance	IOxfReactive	In
state	char*	In

Operation name: notifyStateExited

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyStateExited(IOxfReactive instance,char* state)
Return Type: void
Description: Notify that a state was exited

Argument information for Operation notifyStateExited

Name	Type	Direction
instance	IOxfReactive	In
state	char*	In

Operation name: setThreadName

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: setThreadName(void * osThread,char* name)
Return Type: void
Description: Set the animation thread name

Argument information for Operation setThreadName

Name	Type	Direction
osThread	void *	In
name	char*	In

Operation name: switchOSThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: switchOSThread(void * oldThread,void * newThread)
Return Type: void
Description: Switch the thread used by an implementation of IOxfActive (typically called by the default active class on the event loop initialization).

Argument information for Operation switchOSThread

Name	Type	Direction
oldThread	void *	In
newThread	void *	In

Operation name: notifyTerminateConnector

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyTerminateConnector(IOxfReactive instance,char* connector)
Return Type: void
Description: Notify that the reactive instance reached a terminate connector

Argument information for Operation notifyTerminateConnector

Name	Type	Direction
instance	IOxfReactive	In
connector	char*	In

Operation name: notifyTimeoutCancelled

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyTimeoutCancelled(IOxfTimeout tm)
Return Type: void
Description: Notify that a timeout was canceled

Argument information for Operation notifyTimeoutCancelled

Name	Type	Direction
tm	IOxfTimeout	In

Operation name: notifyTimeoutSet

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyTimeoutSet(IOxfTimeout tm)
Return Type: void
Description: Notify that a timeout was scheduled

Argument information for Operation notifyTimeoutSet

Name	Type	Direction
tm	IOxfTimeout	In

Operation name: deregisterForeignThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: deregisterForeignThread(void * osThread)
Return Type: void
Description: de-register an external thread (not related to active instances) that was registered on the animation

Argument information for Operation deregisterForeignThread

Name	Type	Direction
osThread	void *	In

Operation name: registerForeignThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: registerForeignThread(void * osThread,char* name)
Return Type: void
Description: register an external thread (not related to active instances) that was registered on the animation

Argument information for Operation registerForeignThread

Name	Type	Direction
osThread	void *	In
name	char*	In

Operation name: notifyError

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyError(char* msg)
Return Type: void
Description: Nofiy the animation that an error occured

Argument information for Operation notifyError

Name	Type	Direction
msg	char*	In

Operation name: notifyError

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyError(void * context,char* msg)
Return Type: void
Description: Nofiy the animation that an error occured

Argument information for Operation notifyError

Name	Type	Direction
context	void *	In
msg	char*	In

Operation name: notifySendingEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifySendingEvent(IOxfEvent ev,IOxfEventGenerationParams params)
Return Type: void
Description: Notify that an event is being send

Argument information for Operation notifySendingEvent

Name	Type	Direction
ev	IOxfEvent	In
params	IOxfEventGenerationParams	In

Operation name: notifyThreadCreated

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyThreadCreated(IOxfActive context,void * osThread)
Return Type: void
Description: Notify that a thread was created

Argument information for Operation notifyThreadCreated

Name	Type	Direction
context	IOxfActive	In
osThread	void *	In

Operation name: notifyThreadDestroyed

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyThreadDestroyed(void * osThread)
Return Type: void
Description: Notify that a thread is about to be destroyed

Argument information for Operation notifyThreadDestroyed

Name	Type	Direction
osThread	void *	In

Operation name: notifyEventCancelled

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyEventCancelled(void * osThread,IOxfEvent ev)
Return Type: void
Description: Notify that an event was canceled (required for direct reactive deletion)

Argument information for Operation notifyEventCancelled

Name	Type	Direction
osThread	void *	In
ev	IOxfEvent	In

Operation name: resetCallStack

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: resetCallStack(void * osThread)
Return Type: void
Description: Reset the call stack associated with the specified thread

Argument information for Operation resetCallStack

Name	Type	Direction
osThread	void *	In

Operation name: notifyIdle

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyIdle(void * osThread)
Return Type: void
Description: Notify that the specified thread is idle (no events to consume)

Argument information for Operation notifyIdle

Name	Type	Direction
osThread	void *	In

Operation name: notifyReady

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyReady(void * osThread)
Return Type: void
Description: Notify that the specified thread is ready to run.
This is used by the animation to perform function calls on the context of the thread before continue the event loop.

Argument information for Operation notifyReady

Name	Type	Direction
osThread	void *	In

Operation name: notifyFirstStep

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyFirstStep(void * osThread)
Return Type: void
Description: Notify that the first step was performed (e.g. OXF::initialize()) in the context of the specified thread.

Argument information for Operation notifyFirstStep

Name	Type	Direction
osThread	void *	In

Operation name: init

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: init(int argc,char** argv,unsigned int defaultPort,char* defaultHost)
Return Type: bool
Description: Initialize the animation
Return true if the initialization was sucessful
Animated applications should terminate if the return value is false.

Argument information for Operation init

Name	Type	Direction
argc	int	In
argv	char**	In
defaultPort	unsigned int	In
defaultHost	char*	In

Operation name: notifyEventGetBegin

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyEventGetBegin(void * osThread)
Return Type: void
Description: Notify that the specified thread is about remove an event from the queue

Argument information for Operation notifyEventGetBegin

Name	Type	Direction
osThread	void *	In

Operation name: notifyFrameworkStarted

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyFrameworkStarted()
Return Type: void
Description: Notify the AOM that the framework default message loop isactivated

Operation name: getAnimInstance

Initializer:
Const: false

Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: getAnimInstance(void * inst)
Return Type: AOMInstance
Description: Return the animation instance that is associated with the specified "real" instance

Argument information for Operation getAnimInstance

Name	Type	Direction
inst	void *	In

Operation name: addState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: addState(AOMSState states,char* state)
Return Type: void
Description: Add the state information to the AOM state vector

Argument information for Operation addState

Name	Type	Direction
states	AOMSState	InOut
state	char*	In

Operation name: addTerminateState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: addTerminateState(AOMSState states,bool shouldTerminate)
Return Type: void
Description: Add the state information to the AOM state vector

Argument information for Operation addTerminateState

Name	Type	Direction
states	AOMSState	InOut
shouldTerminate	bool	In

Operation name: notifyStateEntered

Initializer:
Const: false
Trigger: false

Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyStateEntered(AOMInstance instance,char* state)
Return Type: void
Description: Notify that a state was entered

Argument information for Operation notifyStateEntered

Name	Type	Direction
instance	AOMInstance	In
state	char*	In

Operation name: notifyStateExited
Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyStateExited(AOMInstance instance,char* state)
Return Type: void
Description: Notify that a state was exited

Argument information for Operation notifyStateExited

Name	Type	Direction
instance	AOMInstance	In
state	char*	In

Operation name: shouldNotifyIdle
Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: shouldNotifyIdle(void * osThread)
Return Type: bool
Description: Check if the specified thread should become idle.
If this operation returns true notifyIdle() should be called otherwise you should wait for events and then call notifyReady()

Argument information for Operation shouldNotifyIdle

Name	Type	Direction
osThread	void *	In

Operation name: registerReactiveInstance
Initializer:
Const: false

Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: registerReactiveInstance(IOxfReactive instance,IOxfAnimReactive proxy)
Return Type: void
Description: Register a reactive instance with its animation proxy

Argument information for Operation registerReactiveInstance

Name	Type	Direction
instance	IOxfReactive	In
proxy	IOxfAnimReactive	In

Operation name: notifyEventPutEnd

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyEventPutEnd(void * osThread,IOxfEvent ev)
Return Type: void
Description: Notify that the event was put into the thread queue

Argument information for Operation notifyEventPutEnd

Name	Type	Direction
osThread	void *	In
ev	IOxfEvent	In

Operation name: notifyEventPutBegin

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyEventPutBegin(void * osThread,IOxfEvent ev,bool blocking)
Return Type: void
Description: Notify that the event is about to be put into the thread queue

Argument information for Operation notifyEventPutBegin

Name	Type	Direction
osThread	void *	In
ev	IOxfEvent	In
blocking	bool	In

Operation name: notifyEventGetEnd

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyEventGetEnd(void * osThread,IOxfEvent ev)
Return Type: void
Description: Notify that the specified thread removed the event from the queue

Argument information for Operation notifyEventGetEnd

Name	Type	Direction
osThread	void *	In
ev	IOxfEvent	In

Operation name: notifyHandleEventEnd

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyHandleEventEnd(IOxfReactive instance,IOxfEvent ev)
Return Type: void
Description: Notify that an event was handled

Argument information for Operation notifyHandleEventEnd

Name	Type	Direction
instance	IOxfReactive	In
ev	IOxfEvent	In

Operation name: notifyStartBehaviorEnd

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyStartBehaviorEnd(IOxfReactive instance)
Return Type: void
Description: Notify that the behavior (i.e. statemachine) of a reactive instance is completed.

Argument information for Operation notifyStartBehaviorEnd

Name	Type	Direction
instance	IOxfReactive	In

Operation name: notifyFrameworkEventCreated

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyFrameworkEventCreated(IOxfEvent ev)
Return Type: void
Description: Notify that a framework animated event was created

Argument information for Operation notifyFrameworkEventCreated

Name	Type	Direction
ev	IOxfEvent	In

Operation name: registerAnimThreadManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: registerAnimThreadManager(IOxfAnimThreadManager manager)
Return Type: void
Description: Register the threads manager

Argument information for Operation registerAnimThreadManager

Name	Type	Direction
manager	IOxfAnimThreadManager	In

Operation name: getThreadManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: getThreadManager()
Return Type: IOxfAnimThreadManager
Description: Return the threads manager

Operation name: getTimerManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public

Signature: getTimerManager()
Return Type: IOxfAnimTimerManager
Description: Return the timer manager

Operation name: notifyEventStep

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyEventStep(void * osThread)
Return Type: void
Description: Notify that an event step had occured

Argument information for Operation notifyEventStep

Name	Type	Direction
osThread	void *	In

Operation name: setThreadContext

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: setThreadContext(IOxfActive oxfContext,AOMInstance instance)
Return Type: void
Description: Set the animation thread name by an animation instance

Argument information for Operation setThreadContext

Name	Type	Direction
oxfContext	IOxfActive	In
instance	AOMInstance	In

Operation name: registerAnimHelper

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: registerAnimHelper(IOxfAnimHelper animHelper)
Return Type: void
Description: register the singleton implementation of IOxfAnimHelper

Argument information for Operation registerAnimHelper

Name	Type	Direction
animHelper	IOxfAnimHelper	In

Operation name: notifyReactiveInstanceDeleted

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyReactiveInstanceDeleted(IOxfReactive instance)
Return Type: void
Description: Notify that a reactive instance is deleted

Argument information for Operation notifyReactiveInstanceDeleted

Name	Type	Direction
instance	IOxfReactive	In

Operation name: cleanupOnEndApplication

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: cleanupOnEndApplication()
Return Type: void
Description: Perform explicit AOM cleanup
Required to cleanup the AOM in adapters that do not support automatic cleanup

Package: omcom

Class Information for Package: omcom

Class name: OMSData

Description:The base instrumentation message
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Package: oxf

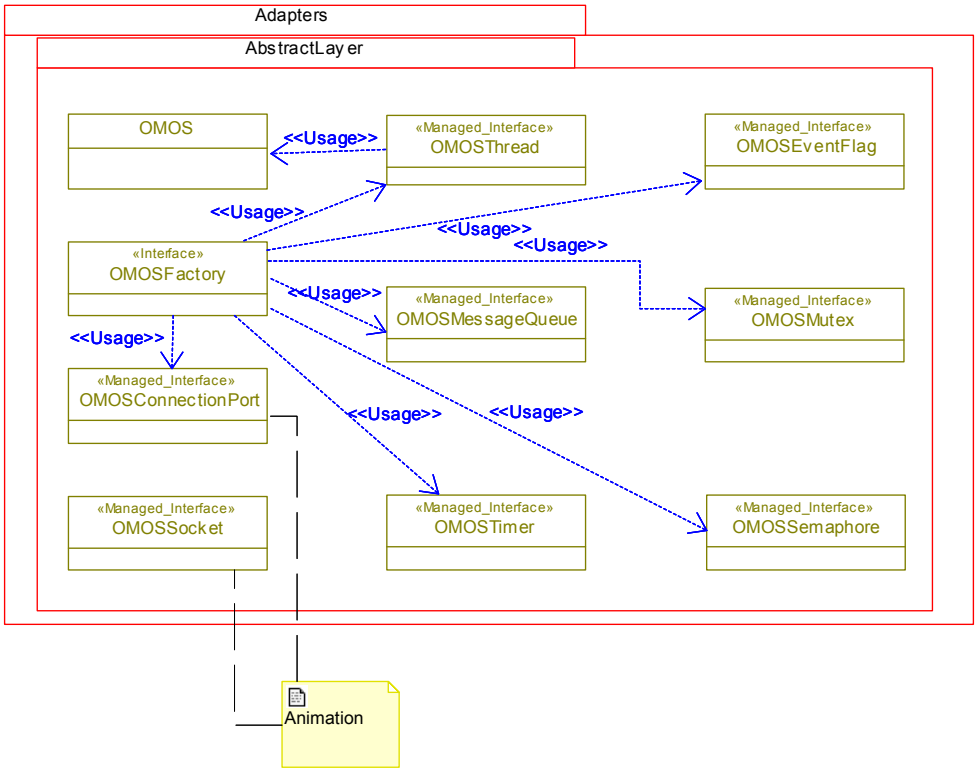
Package Information

Description: The Object eXecution Framework

Package: Adapters

Object Model Diagram Information

Object Model Diagram name: Generic adapter
Description: Overview of the OSAL interface



Package Information
Description: The Adapters package contains the OSAL (OS Abstraction Layer).

Package: AbstractLayer

Class Information for Package: AbstractLayer

Class name: OMOSMessageQueue

Description:Message queue abstraction.
The queue is expected to pass pointers to the messages rather than the data itself.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOSMessageQueue

Operation name: get

Initializer:
Const: false

Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: get()
Return Type: void *
Description: Get a message from the queue.
This is not a blocking call, it should return 0 if the queue is empty

Operation name: getMessageList

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getMessageList(OMList<void*> & l)
Return Type: void
Description: copy the messages in the queue into l

Argument information for Operation getMessageList

Name	Type	Direction
l	OMList<void*> & %s	InOut

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: get the real OS element

Operation name: isEmpty

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: isEmpty()
Return Type: int
Description: check if the queue is empty

Operation name: isFull

Initializer:
Const: false

Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: isFull()
Return Type: bool
Description: check if the queue is full

Operation name: pend

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: pend()
Return Type: void
Description: block until there are messages in the queue

Operation name: putMessage

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: putMessage(void * m,IOxfEventGenerationParams params)
Return Type: bool
Description: put a message to the queue

Argument information for Operation putMessage

Name	Type	Direction
m	void *	In
params	IOxfEventGenerationParams	In

Operation name: setOwnerProcess

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: setOwnerProcess(void * /* process */)
Return Type: void
Description: set the thread that owns the queue (required in some adapters)

Argument information for Operation setOwnerProcess

Name	Type	Direction
/* process */	void *	In

Operation name: ~OMOSMessageQueue

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSMessageQueue()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: put

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: put(void * /* msg */,bool /* fromISR */)
Return Type: bool
Description: Add a message to the queue

Argument information for Operation put

Name	Type	Direction
/* msg */	void *	In
/* fromISR */	bool	In

Class name: OMOSEventFlag

Description:Event flag abstraction.
Used for execution synchronization between threads.
Initially the event flag should have no tokens - if the first call is to wait(), the caller should block.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOSEventFlag

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: get the real OS element

Operation name: reset

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: reset()
Return Type: void
Description: Reset the event flag.
All waiting threads should be signaled.
The next call to wait() should block

Operation name: signal

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: signal()
Return Type: void
Description: Signal the first waiting thread.
If there are no threads waiting, increase the token count (following call to wait() pass without blocking).

Operation name: wait

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: wait(int t)
Return Type: void
Description: Wait for a token or a timeout.

Argument information for Operation wait

Name	Type	Direction
t	int	In

Operation name: ~OMOSEventFlag

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSEventFlag()
Return Type:

Description: Virtual destructor to enable polymorphic deletion

Class name: OMOSThread

Description:Thread abstraction.
Either creates a new thread or represents an existing thread.
When creating a new thread it should be blocked until the call to start().
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMOSThread

Attribute Name: DefaultMessageQueueSize

Default Value: 100
Static: true
Visibility: public
Type: long
Stereotype:
Description: The default message queue size

Attribute Name: DefaultStackSize

Default Value: 1000
Static: true
Visibility: public
Type: long
Stereotype:
Description: The default thread stack-size

Attribute Name: DefaultThreadPriority

Default Value: 0
Static: true
Visibility: public
Type: long
Stereotype:
Description: The default thread priority

Operation information for Class: OMOSThread

Operation name: CleanupCommunicationLayer

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: CleanupCommunicationLayer()
Return Type: void
Description: Cleanup IP resources allocated for the thread (required by some adapters).

Operation name: exeOnMyThread

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: exeOnMyThread()
Return Type: bool
Description: the following service returns true iff it is invoked from the same os thread as the on the object represents

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: get the real OS element

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getOsHandle(void * osHandle)
Return Type: void *
Description: pass the real OS element to the osHandle

Argument information for Operation getOsHandle

Name	Type	Direction
osHandle	void *	InOut

Operation name: getThreadEndClbk

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getThreadEndClbk(EndCallBack clb_p,void * arg1_p,bool onExecuteThread)
Return Type: void
Description: get the thread termination call

Argument information for Operation getThreadEndClbk

Name	Type	Direction

clb_p	EndCallBack	Out
arg1_p	void *	Out
onExecuteThread	bool	In

Operation name: InitCommunicationLayer

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: InitCommunicationLayer()
Return Type: void
Description: Initialize IP resources allocated for the thread (required by some adapters).

Operation name: resume

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: resume()
Return Type: void
Description: resume suspended thread

Operation name: setEndOSThreadInDtor

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: setEndOSThreadInDtor(bool val)
Return Type: void
Description: Mark the thread as under destruction

Argument information for Operation setEndOSThreadInDtor

Name	Type	Direction
val	bool	In

Operation name: setPriority

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false

Visibility: public
Signature: setPriority(int pr)
Return Type: void
Description: set the thread priority

Argument information for Operation setPriority

Name	Type	Direction
pr	int	In

Operation name: start

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: start()
Return Type: void
Description: Start the thread execution

Operation name: suspend

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: suspend()
Return Type: void
Description: Suspend the thread

Operation name: ~OMOSThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSThread()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Type information for Class OMOSThread

Type name: EndCallBack

Description: The end thread callback type
Kind: Language
Declaration: typedef void (*%s)(void*)

Type name: OMOSThreadEndCallBack

Description: alias of EndCallBack
Kind: Typedef
Basic Type: EndCallBack
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Class name: OMOSMutex

Description:Mutex (binary semaphore) abstraction.
The mutex must allow recursive lock policy.
This means that once a thread obtained the mutex it can call lock() any number of times and pass.
The mutex is released only when the number of unlock() calls is equal to the number of lock() calls done by the owner thread.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOSMutex

Operation name: free

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: free()
Return Type: void
Description: backward compatibility support for non OSE applications

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: get the OS implementation

Operation name: lock

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public

Signature: lock()
Return Type: void
Description: obtain the mutex

Operation name: unlock

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: unlock()
Return Type: void
Description: release the mutex

Operation name: ~OMOSMutex

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSMutex()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Class name: OMOSSemaphore

Description:Semaphore abstraction.
This class is not used by the framework.
It is provided for the completeness of the OS abstraction.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOSSemaphore

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: get the real OS element

Operation name: signal

Initializer:
Const: false

Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: signal()
Return Type: void
Description: signal the waiting threads

Operation name: wait

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: wait(long timeout)
Return Type: bool
Description: wait on the semaphore

Argument information for Operation wait

Name	Type	Direction
timeout	long	In

Operation name: ~OMOSSemaphore

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSSemaphore()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Class name: OMOSTimer

Description:A tick-timer abstraction.
Represents a real-time timer that ticks for a specific duration, or an idle timer that ticks only when the system is idle.
The idle timer is used for the time simulation feature.

Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOSTimer

Operation name: getOsHandle

Initializer:
Const: true

Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: get the real OS element

Operation name: ~OMOSTimer

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSTimer()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Class name: OMOS

Description:Utility class for general RTOS services
The implementation of these services is done at the adapter level by writing the operation bodies.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOS

Operation name: endApplication

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: endApplication(int errorCode)
Return Type: void
Description: os-specific actions to take at the end of OXFInit
after the environment is set (i.e. main thread, timer etc)
and before return from the function

Argument information for Operation endApplication

Name	Type	Direction
errorCode	int	In

Operation name: endProlog

Initializer:
Const: false
Trigger: false

Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: endProlog()
Return Type: void
Description: Called just before terminating the application.
Allow adapter to specific tasks that should be done before termination.

Operation name: initEpilog

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: initEpilog()
Return Type: void
Description: Called at the end of OXF::init() to allow adapter specific initializations.

Class name: OMOSSocket

Description:Client socket abstraction
Used for animation
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOSSocket

Operation name: Create

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: Create(char* SocketAddress,unsigned int nSocketPort)
Return Type: int
Description: Create the socket.
Return the socket discriminator or 0 on error

Argument information for Operation Create

Name	Type	Direction
SocketAddress	char*	In
nSocketPort	unsigned int	In

Operation name: Send

Initializer:
Const: false

Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: Send(char* lpBuf,int nBufLen)
Return Type: int
Description: Send a message via the socket (blocking)

Argument information for Operation Send

Name	Type	Direction
lpBuf	char*	In
nBufLen	int	In

Operation name: Receive

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: Receive(char* lpBuf,int nBufLen)
Return Type: int
Description: Receive a message from the socket (blocking)

Argument information for Operation Receive

Name	Type	Direction
lpBuf	char*	In
nBufLen	int	In

Operation name: Close

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: Close()
Return Type: void
Description: close the socket

Operation name: ~OMOSSocket

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSSocket()

Return Type:
Description: Virtual destructor to enable polymorphic deletion

Class name: OMOSFactory

Description:OS elements creation factory
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOSFactory

Operation name: createOMOSEventFlag

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: createOMOSEventFlag()
Return Type: OMOSEventFlag
Description: Create the adapter implementation of the event flag

Operation name: createOMOSIdleTimer

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: createOMOSIdleTimer(void (callback)(void*),void * param)
Return Type: OMOSTimer
Description: Create the adapter implementation of the idle timer (an idle timer send ticks only when the system is idle)

Argument information for Operation createOMOSIdleTimer

Name	Type	Direction
callback	void (%s)(void*)	In
param	void *	In

Operation name: createOMOSMessageQueue

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: createOMOSMessageQueue(bool shouldGrow,long messageQueueSize)
Return Type: OMOSMessageQueue
Description: Create the adapter implementation of the message queue

Argument information for Operation createOMOSMessageQueue

Name	Type	Direction
shouldGrow	bool	In
messageQueueSize	long	In

Operation name: createOMOSMutex

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: createOMOSMutex()
Return Type: OMOSMutex
Description: Create the adapter implementation of the mutex

Operation name: createOMOSSemaphore

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: createOMOSSemaphore(unsigned long semFlags,unsigned long initialCount,unsigned long maxCount,const char * const name)
Return Type: OMOSSemaphore
Description: Create the adapter implementation of the semaphore

Argument information for Operation createOMOSSemaphore

Name	Type	Direction
semFlags	unsigned long	In
initialCount	unsigned long	In
maxCount	unsigned long	In
name	const char * const %s	In

Operation name: createOMOSThread

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: createOMOSThread(void (entry)(void*),void * param,const char * const threadName,long stackSize)
Return Type: OMOSThread
Description: Create the adapter implementation of the thread

Argument information for Operation createOMOSThread

Name	Type	Direction

entry	void (%s)(void*)	In
param	void *	In
threadName	const char * const %s	In
stackSize	long	In

Operation name: createOMOSTickTimer

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: createOMOSTickTimer(OxfTimeUnit time,void (callback)(void*),void * param)
Return Type: OMOSTimer
Description: Create the adapter implementation of the real-time timer

Argument information for Operation createOMOSTickTimer

Name	Type	Direction
time	OxfTimeUnit	In
callback	void (%s)(void*)	In
param	void *	In

Operation name: createOMOSWrapperThread

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: createOMOSWrapperThread(void * osHandle)
Return Type: OMOSThread
Description: Create the adapter implementation of the wrapper thread (an OMOSThread representation of a thread that already exist in the system).

Argument information for Operation createOMOSWrapperThread

Name	Type	Direction
osHandle	void *	In

Operation name: delayCurrentThread

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: delayCurrentThread(OxfTimeUnit ms)
Return Type: void
Description: Make the current thread delay (blocking) for the specified time.

Argument information for Operation delayCurrentThread

Name	Type	Direction
ms	OxfTimeUnit	In

Operation name: getCurrentThreadHandle

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getCurrentThreadHandle()
Return Type: void *
Description: return the current thread OS handle (id)

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance()
Return Type: OMOSFactory
Description: Create the OSFactory (replaces V3.0 global function theOSFactory())

Operation name: waitOnThread

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: waitOnThread(void * osHandle,OxfTimeUnit ms)
Return Type: bool
Description: wait for a thread to terminate

Argument information for Operation waitOnThread

Name	Type	Direction
osHandle	void *	In
ms	OxfTimeUnit	In

Operation name: createOMOSConnectionPort

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false

Visibility: public
Signature: createOMOSConnectionPort()
Return Type: OMOSConnectionPort
Description: Create the adapter implementation of the connection port

Operation name: ~OMOSFactory

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSFactory()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Class name: OMOSConnectionPort

Description:This class provides the animation messaging acknowledge protocol with Rhapsody and a wrapper on the socket class.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOSConnectionPort

Operation name: Connect

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: Connect(char* SocketAddress,unsigned int nSocketPort)
Return Type: int
Description: Connect to the specified address and port

Argument information for Operation Connect

Name	Type	Direction
SocketAddress	char*	In
nSocketPort	unsigned int	In

Operation name: Send

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: Send(OMSData m)

Return Type: int
Description: Send the data

Argument information for Operation Send

Name	Type	Direction
m	OMSData	In

Operation name: ~OMOSConnectionPort

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSConnectionPort()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: SetDispatcher

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: SetDispatcher(void dispfunc(OMSData*))
Return Type: void
Description: Set the message handler for incoming messages.

Argument information for Operation SetDispatcher

Name	Type	Direction
dispfunc	void %s(OMSData*)	In

Attribute information for Package AbstractLayer

Attribute name: MAX_LEN_STR

Type: int
Stereotype:
Declaration:
Default Value: 6
Description: Used by the connection port implementation

Package: Anim

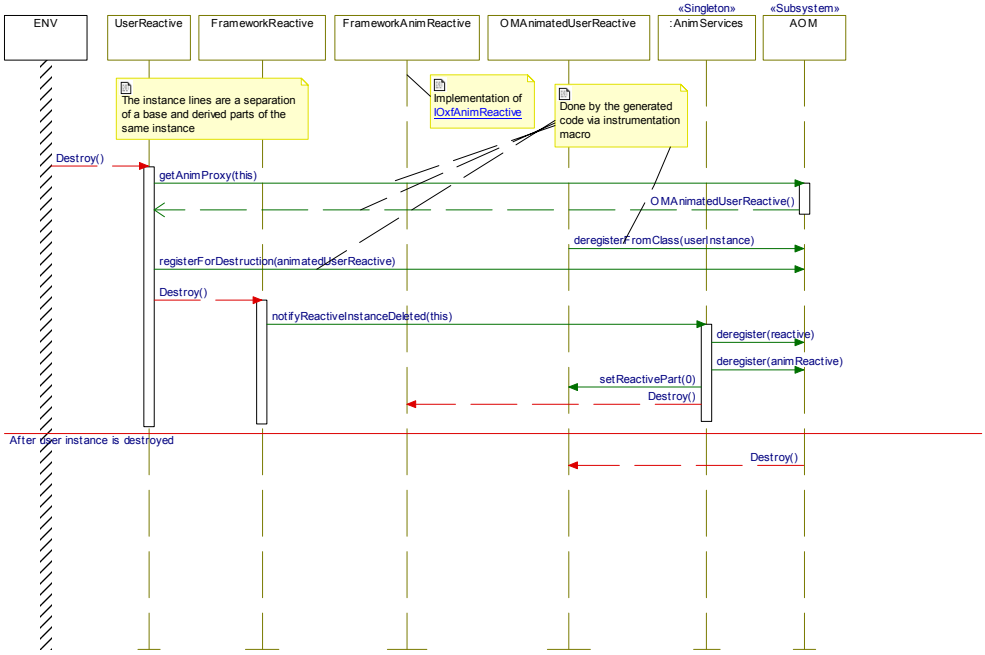
Package Information
Description: Animation support package

Package: AnimAPI

Sequence Diagram Information

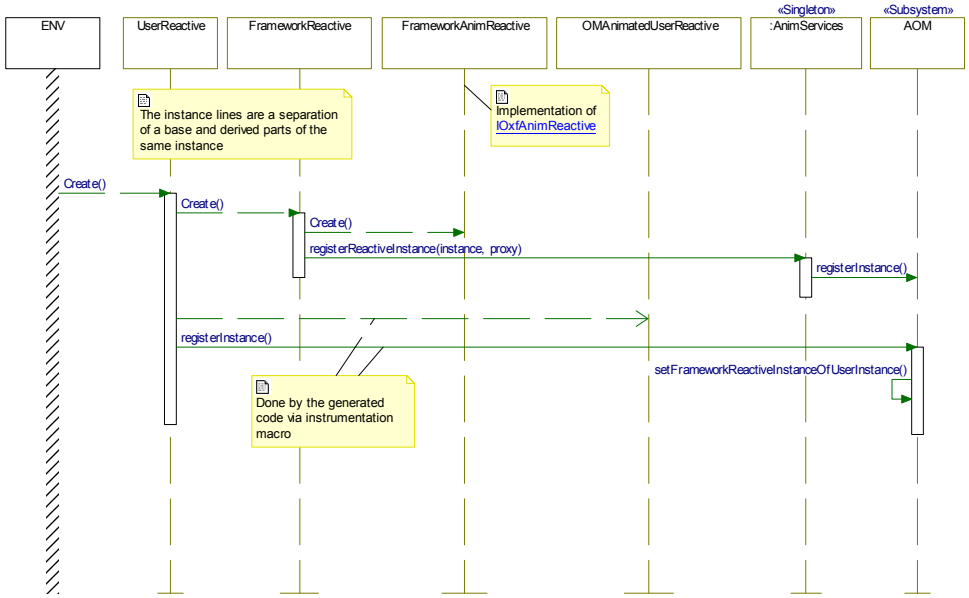
Sequence Diagram name: Destruction of reactive instance

Description: This SD shows the protocol between the OXF and the AnimServices on the destruction of an animated reactive instance



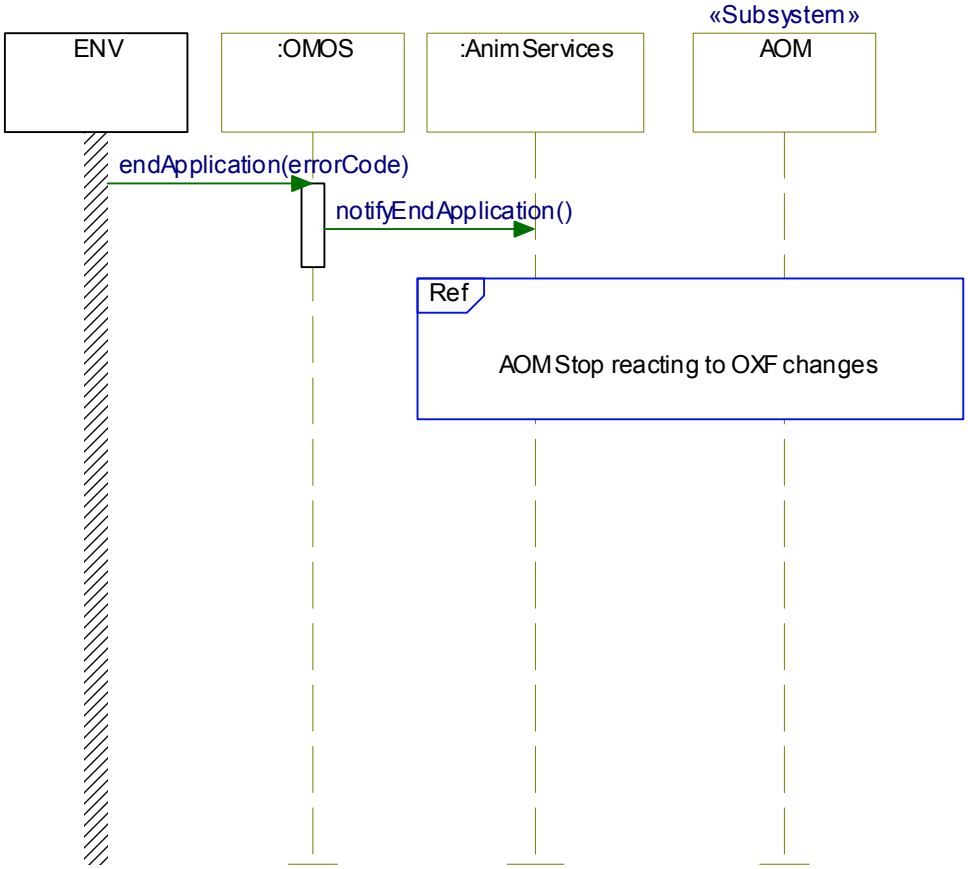
Sequence Diagram name: Creation of reactive instance

Description: This SD shows the protocol between the OXF and the AnimServices on the creation of an animated reactive instance



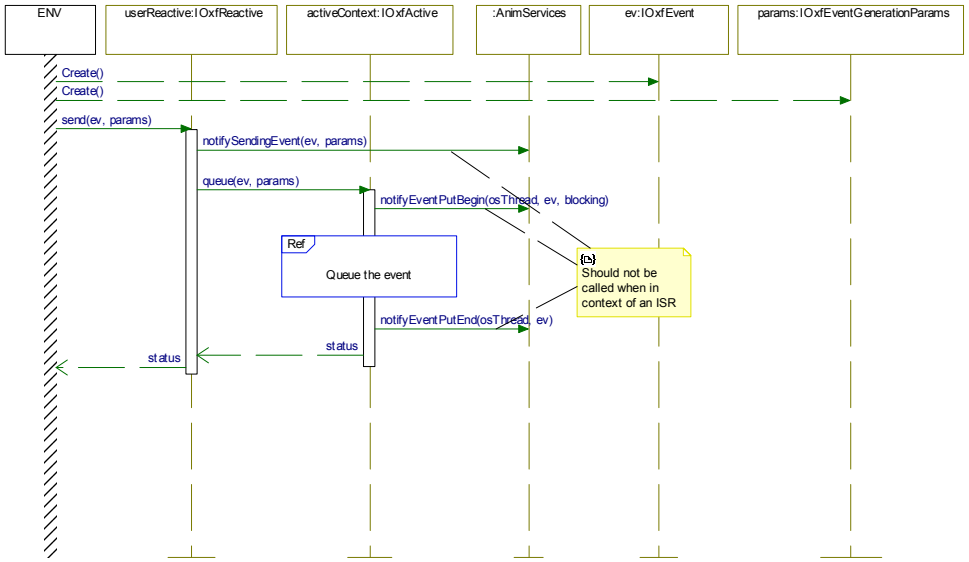
Sequence Diagram name: End of application

Description: This SD shows the protocol between the OXF and the AnimServices on the termination of an animated application



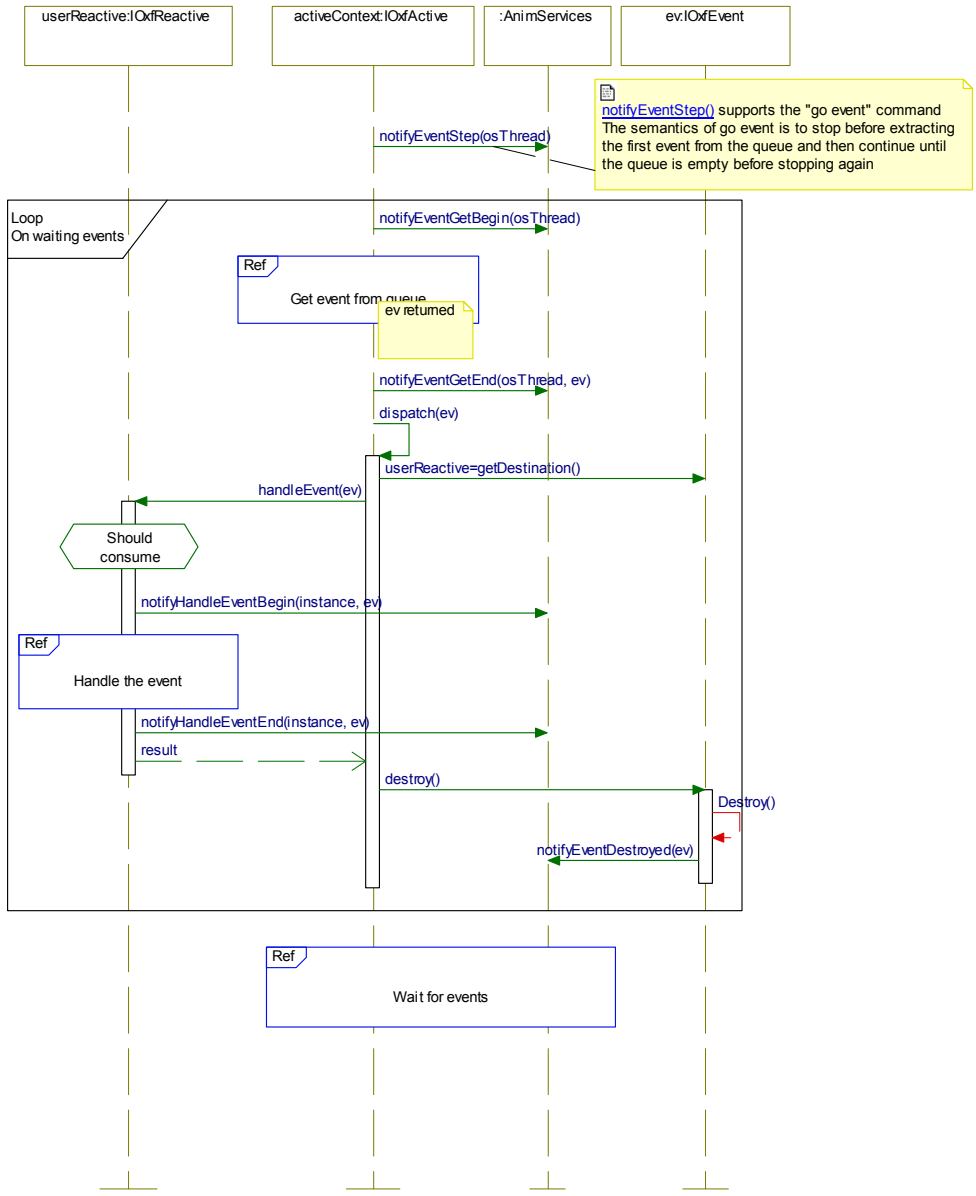
Sequence Diagram name: Event sending

Description: This SD shows the protocol between the OXF and the AnimServices upon the sending of an event



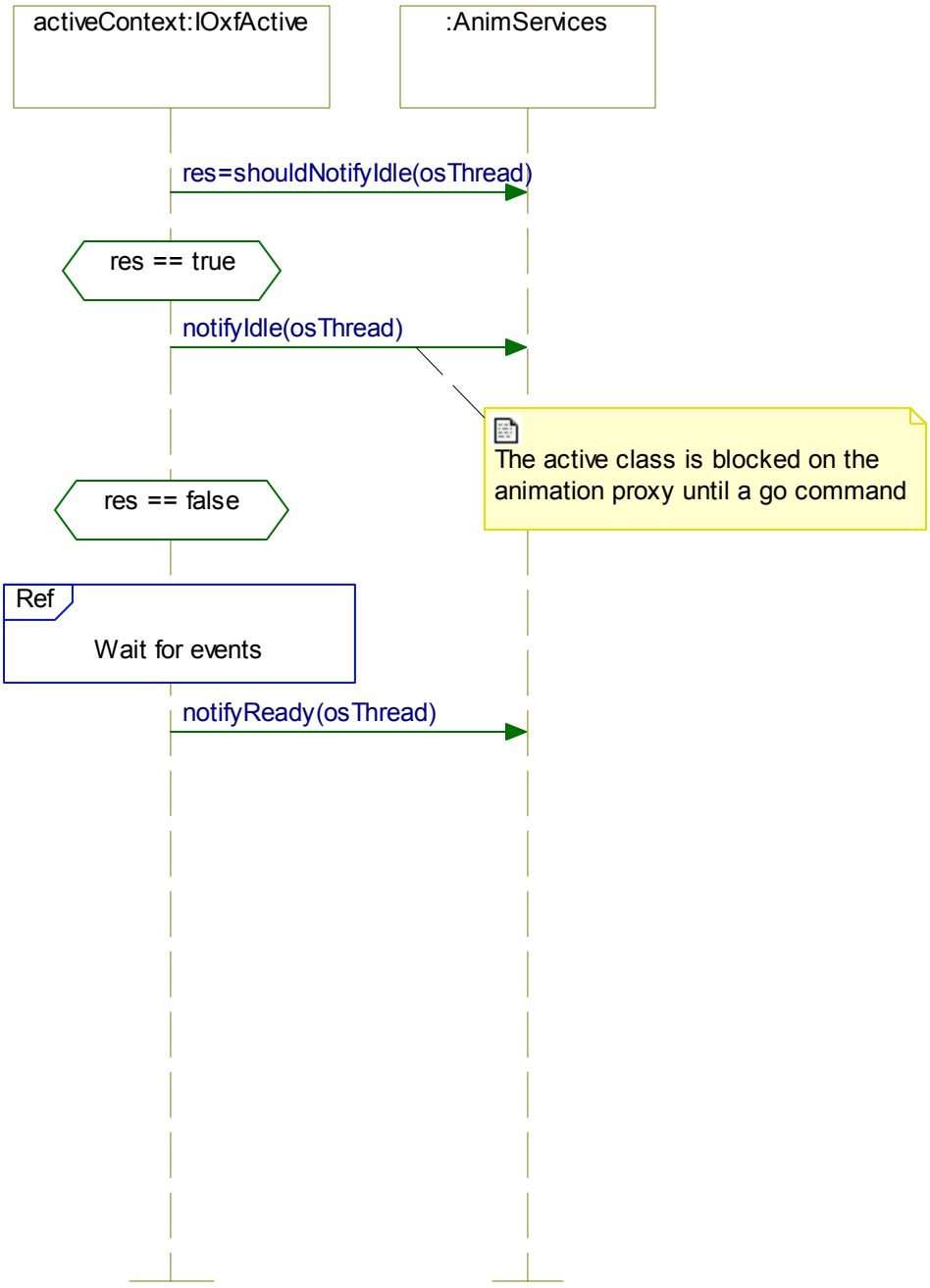
Sequence Diagram name: Event dispatching

Description: This SD shows the protocol between the OXF and the AnimServices upon the dispatch of an event



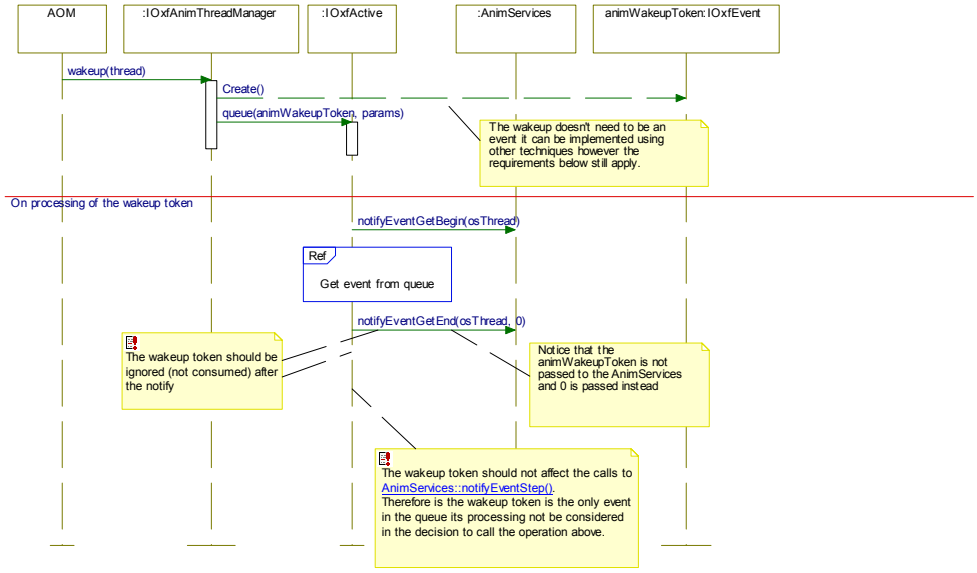
Sequence Diagram name: Wait for events

Description: This SD shows the protocol between the OXF and the AnimServices when an active class waits for events



Sequence Diagram name: Wakeup an active class that is waiting on the queue

Description: This SD shows the protocol between the OXF and AOM on wakeup of an active class



Class Information for Package: AnimAPI

Class name: IOxfAnimReactive

Description:An animation interface to the reactive class implementation
Enables the animation layer to communicate with the reactive class implementation
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: IOxfAnimReactive

Operation name: serializeStates

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: serializeStates(AOMSState states)
Return Type: void
Description: Serializes the states.

Argument information for Operation serializeStates

Name	Type	Direction
states	AOMSState	InOut

Operation name: getOxfReactive

Initializer:
Const: true
Trigger: false
Abstract: true

Static: false
Virtual: false
Visibility: public
Signature: getOxfReactive()
Return Type: IOxfReactive
Description: Returns the "real" reactive part.
Used (for example) to send events

Operation name: canAcceptEvents

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: canAcceptEvents()
Return Type: bool
Description: Checks if the queue can accept a new event (if it is not full)

Operation name: ~IOxfAnimReactive

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfAnimReactive()
Return Type:
Description: Cleanup

Operation name: getContextThread

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getContextThread()
Return Type: void *
Description: get the identifier of the OS thread that is associated with the active context of the reactive instance

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: send(IOxfEvent ev,void * sender)
Return Type: bool

Description: Send an animated event to the reactive context

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In
sender	void *	In

Relation information for Class IOxfAnimReactive

Relation name: oxfReactive

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: oxfReactive
LinkName:
RoleName: oxfReactive
Type: Association
Description:

Name	Inverse	Source	Target
oxfReactive		IOxfAnimReactive	IOxfReactive

Class name: IOxfAnimTimerManager

Description:An animation interface to the timer manager singleton class implementation
Enables the animation layer to communicate with the timer manager implementation
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: IOxfAnimTimerManager

Operation name: suspend

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: suspend()
Return Type: void
Description: Suspends the timer manager

Operation name: resume

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false

Visibility: public
Signature: resume()
Return Type: void
Description: Resumes the timer manager after it was suspended

Operation name: getElapsedTime

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getElapsedTime()
Return Type: OxfTimeUnit
Description: Returns the elapsed time

Operation name: ~IOxfAnimTimerManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfAnimTimerManager()
Return Type:
Description: Cleanup

Operation name: advanceTime

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: true
Visibility: public
Signature: advanceTime()
Return Type: void
Description: advance the system time to the next waiting timeout

Class name: IOxfAnimThreadManager

Description:An animation interface to thethreads manager singleton class implementation.
Enables the animation layer to communicate with the threads manager in order to control threads
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: IOxfAnimThreadManager

Operation name: wakeup

Initializer:
Const: true

Trigger: false
Abstract: true
Static: false
Virtual: true
Visibility: public
Signature: wakeup(IOxfActive thread)
Return Type: void
Description: Wakeup the specified thread (the thread is supposed to be waiting for events)

Argument information for Operation wakeup

Name	Type	Direction
thread	IOxfActive	In

Operation name: ~IOxfAnimThreadManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfAnimThreadManager()
Return Type:
Description: Cleanup

Class name: IOxfAnimHelper

Description:An animation helper interface a singleton helper class that provides animation with a set of services
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: IOxfAnimHelper

Operation name: getFrameworkEventSignature

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: getFrameworkEventSignature(IOxfEvent event,OMString signature)
Return Type: bool
Description: The operation will return false and empty string for user events, and the event signature for animated internal framework events (e.g. start behavior, timeouts)
Returns true if the event was handled (invisible framework events should return true and with an empty signature)

Argument information for Operation getFrameworkEventSignature

Name	Type	Direction
event	IOxfEvent	In

signature	OMString	InOut

Operation name: ~IOxfAnimHelper

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfAnimHelper()
Return Type:
Description: Cleanup

Operation name: isTimeoutEvent

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: isTimeoutEvent(IOxfEvent ev)
Return Type: bool
Description: Return true if the provided event is a timeout

Argument information for Operation isTimeoutEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: isCancelledEvent

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: isCancelledEvent(IOxfEvent ev)
Return Type: bool
Description: Return true if the provided event is cancelled

Argument information for Operation isCancelledEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: getFrameworkEventClassName

Initializer:
Const: true
Trigger: false
Abstract: true

Static: false
Virtual: false
Visibility: public
Signature: getFrameworkEventClassName(IOxfEvent event,OMString className,bool signatureFormat)
Return Type: bool
Description: The operation will return false and an empty string for user events, and the event class name for animated internal framework events (e.g. start behavior, timeouts)
Returns true if the event was handled (invisible framework events should return true and with an empty signature)

Argument information for Operation getFrameworkEventClassName

Name	Type	Direction
event	IOxfEvent	In
className	OMString	InOut
signatureFormat	bool	In

Stereotype information for Package: AnimAPI

Stereotype name: Subsystem

Description:
OfMetaClass: ClassifierRole

Stereotype name: Singleton

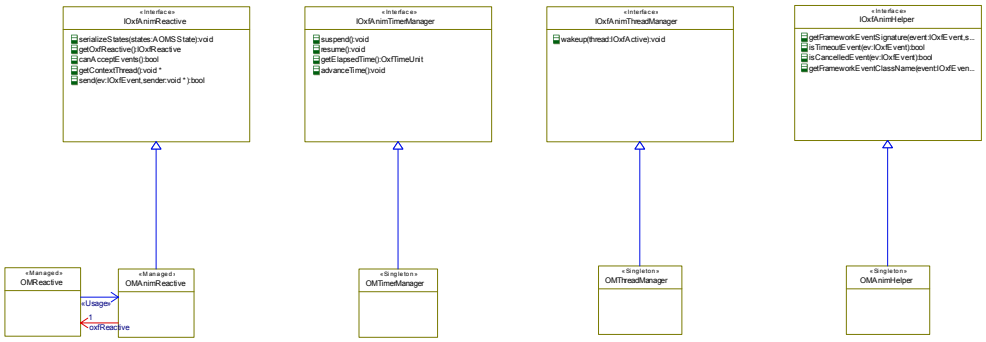
Description:
OfMetaClass: ClassifierRole

Package: AnimImplementation

Object Model Diagram Information

Object Model Diagram name: AnimAPI implementation

Description: This diagram shows the collaboration used to implement the animation API



Class Information for Package: AnimImplementation

Class name: OMAnimReactive

Description:OMReactive animation wrapper
Active: false
Behavior Overridden: false
Composite: false

Reactive: false

Operation information for Class: OMAnimReactive

Operation name: canAcceptEvents

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: canAcceptEvents()
Return Type: bool
Description: Checks if the queue can accept a new event (if it is not full)

Operation name: getOxfReactive

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getOxfReactive()
Return Type: IOxfReactive
Description: Returns the "real" reactive part.
Used (for example) to send events

Operation name: getContextThread

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getContextThread()
Return Type: void *
Description: get the identifier of the OS thread that is associated with the active context of the reactive instance

Operation name: getThread

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: getThread()
Return Type: OMThread
Description: Get the thread

Operation name: OMAnimReactive

Initializer: oxfReactive(context)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMAnimReactive(OMReactive context)
Return Type:
Description: initialization

Argument information for Operation OMAnimReactive

Name	Type	Direction
context	OMReactive	In

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: send(IOxfEvent ev,void * sender)
Return Type: bool
Description: Send an animated event to the reactive context

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In
sender	void *	In

Operation name: serializeStates

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: serializeStates(AOMSState aomArg(states))
Return Type: void
Description: Serializes the states.

Argument information for Operation serializeStates

Name	Type	Direction
aomArg(states)	AOMSState	InOut

Generalization information for Class OMAnimReactive

Generalization name: IOxfAnimReactive

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfAnimReactive	IOxfAnimReactive	OMAnimReactive

Relation information for Class OMAnimReactive

Relation name: oxfReactive

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: oxfReactive
LinkName:
RoleName: oxfReactive
Type: Association
Description: The reactive instance

Name	Inverse	Source	Target
oxfReactive		OMAnimReactive	OMReactive

Class name: OMAnimHelper

Description:OMReactive animation wrapper
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMAnimHelper

Operation name: getFrameworkEventSignature

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getFrameworkEventSignature(IOxfEvent event,OMString signature)
Return Type: bool
Description: he operation will return false and empty string for user events, and the event signature for animated internal framework events (e.g. start behavior, timeouts)
Returns true if the event was handled (invisible framework events should return true and with an empty signature)

Argument information for Operation getFrameworkEventSignature

Name	Type	Direction

event	IOxfEvent	In
signature	OMString	InOut

Operation name: fillTimeoutSignature

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: fillTimeoutSignature(OMTimeout tm,OMString signature)
Return Type: void
Description: Fill the signature for the provided timeout

Argument information for Operation fillTimeoutSignature

Name	Type	Direction
tm	OMTimeout	In
signature	OMString	InOut

Operation name: fillDelaySignature

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: fillDelaySignature(OMTimeout tm,OMString signature)
Return Type: void
Description: Fill the signature for the provided delay-timeout

Argument information for Operation fillDelaySignature

Name	Type	Direction
tm	OMTimeout	In
signature	OMString	InOut

Operation name: fillTime

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: fillTime(OMTimeout tm,OMString str)
Return Type: void
Description: Fill the string with the time for the provided timeout

Argument information for Operation fillTime

Name	Type	Direction
tm	OMTimeout	In
str	OMString	InOut

Operation name: isTimeoutEvent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isTimeoutEvent(IOxfEvent ev)
Return Type: bool
Description: Return true if the provided event is a timeout

Argument information for Operation isTimeoutEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: isCancelledEvent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isCancelledEvent(IOxfEvent ev)
Return Type: bool
Description: Return true if the provided event is cancelled

Argument information for Operation isCancelledEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: getFrameworkEventClassName

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getFrameworkEventClassName(IOxfEvent event,OMString className,bool signatureFormat)
Return Type: bool
Description: The operation will return false and an empty string for user events, and the event class name for animated internal framework events (e.g. start behavior, timeouts)
Returns true if the event was handled (invisible framework events should return true and with an empty signature)

Argument information for Operation getFrameworkEventClassName

Name	Type	Direction
event	IOxfEvent	In
className	OMString	InOut
signatureFormat	bool	In

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance()
Return Type: OMAnimHelper
Description:

Generalization information for Class OMAnimHelper

Generalization name: IOxfAnimHelper

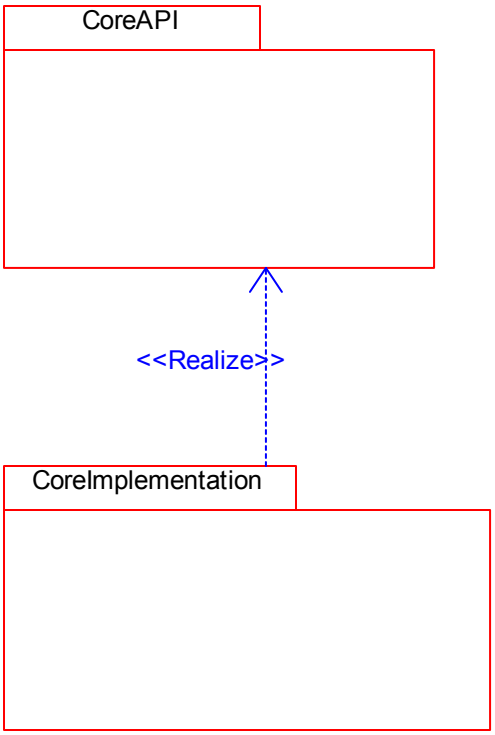
Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfAnimHelper	IOxfAnimHelper	OMAnimHelper

Package: Core

Object Model Diagram Information

Object Model Diagram name: Core packages
Description: The core packages overview



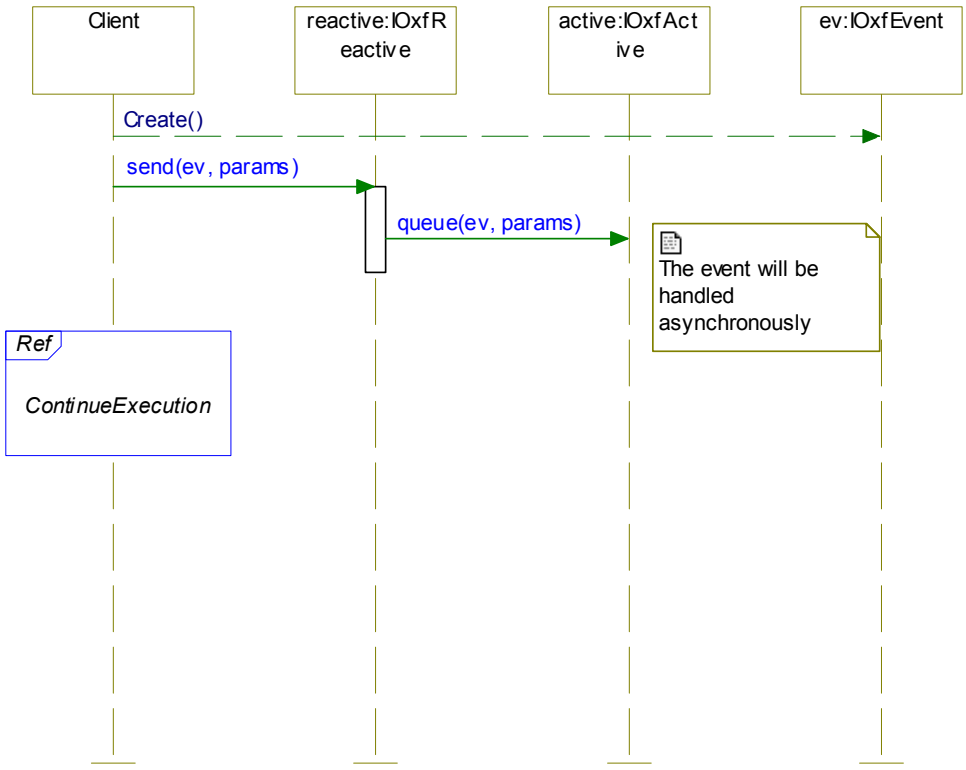
Package Information
Description: The core framework: Multithreaded Event Driven Behavior.

Package: CoreAPI

Sequence Diagram Information

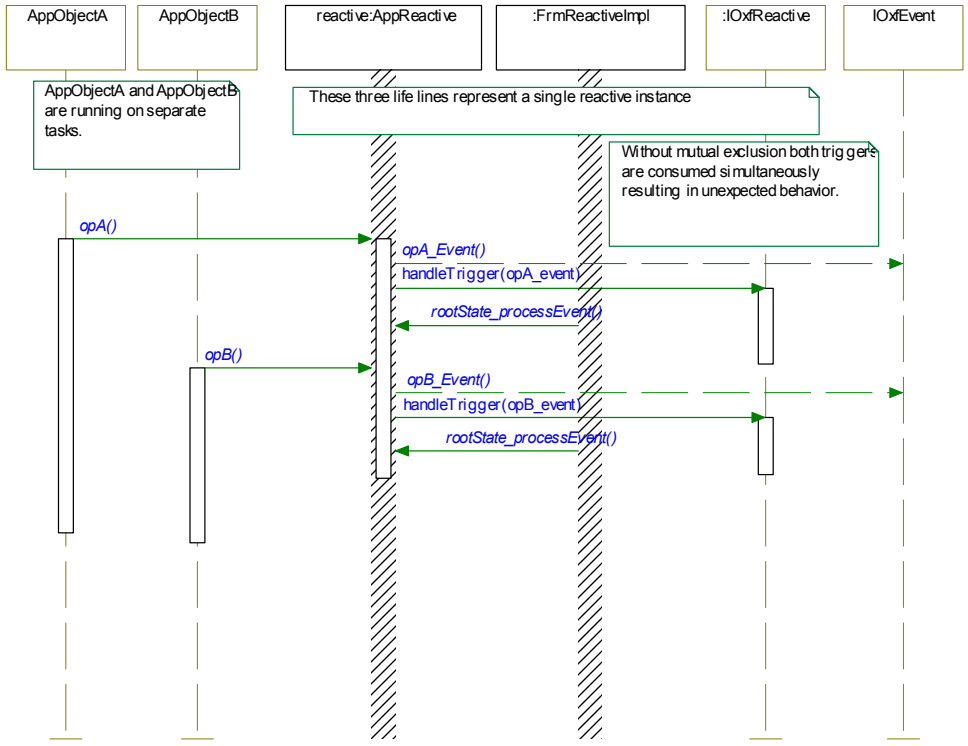
Sequence Diagram name: generating an event

Description: Asynchronous event generation



Sequence Diagram name: Race of Triggered Operation calls

Description: Potential race of Triggered Operation calls

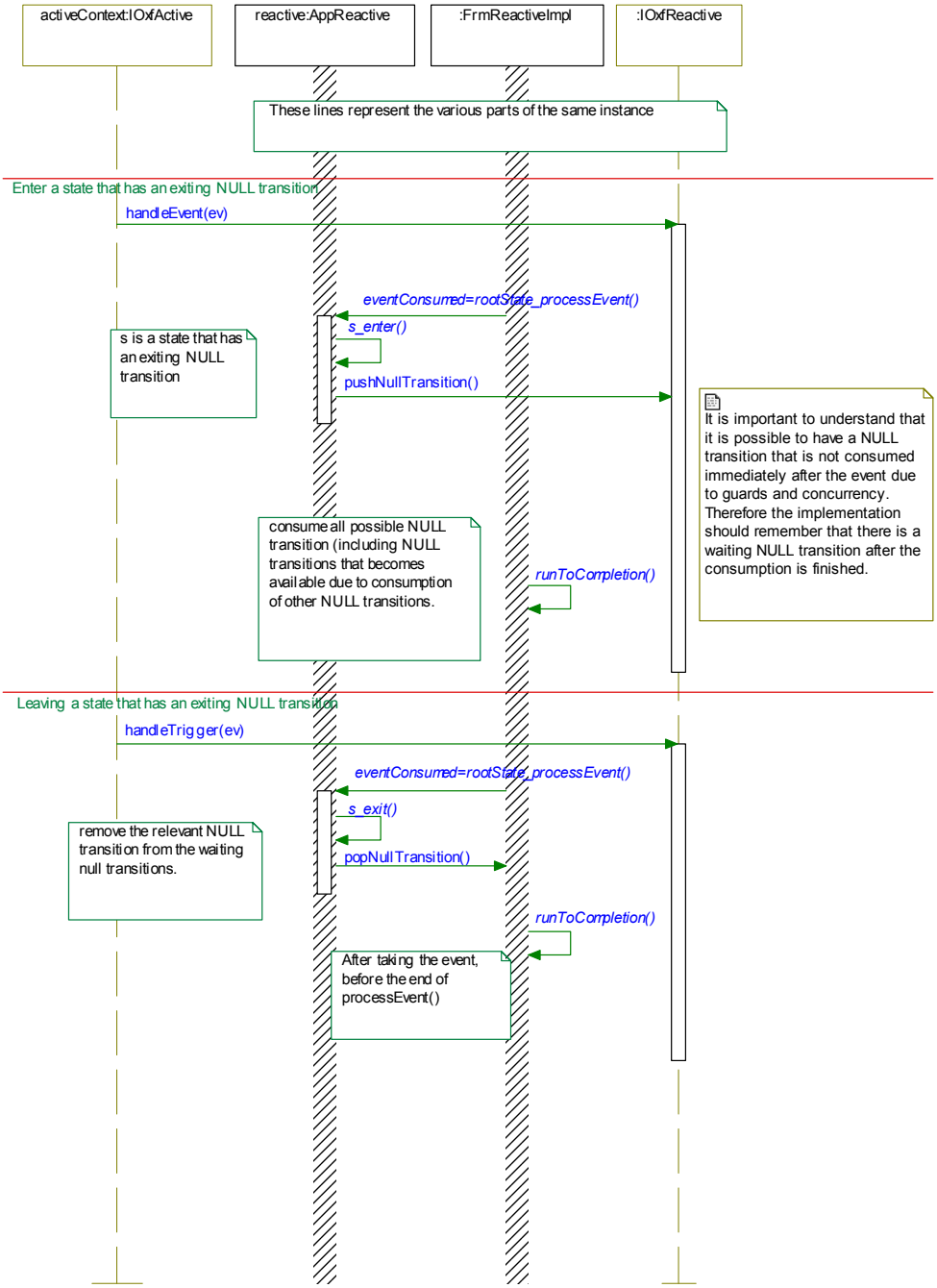


Sequence Diagram name: Consumption of NULL transition

Description: This diagram shows the expected behavior when there are NULL transitions to be consumed as part of an event.

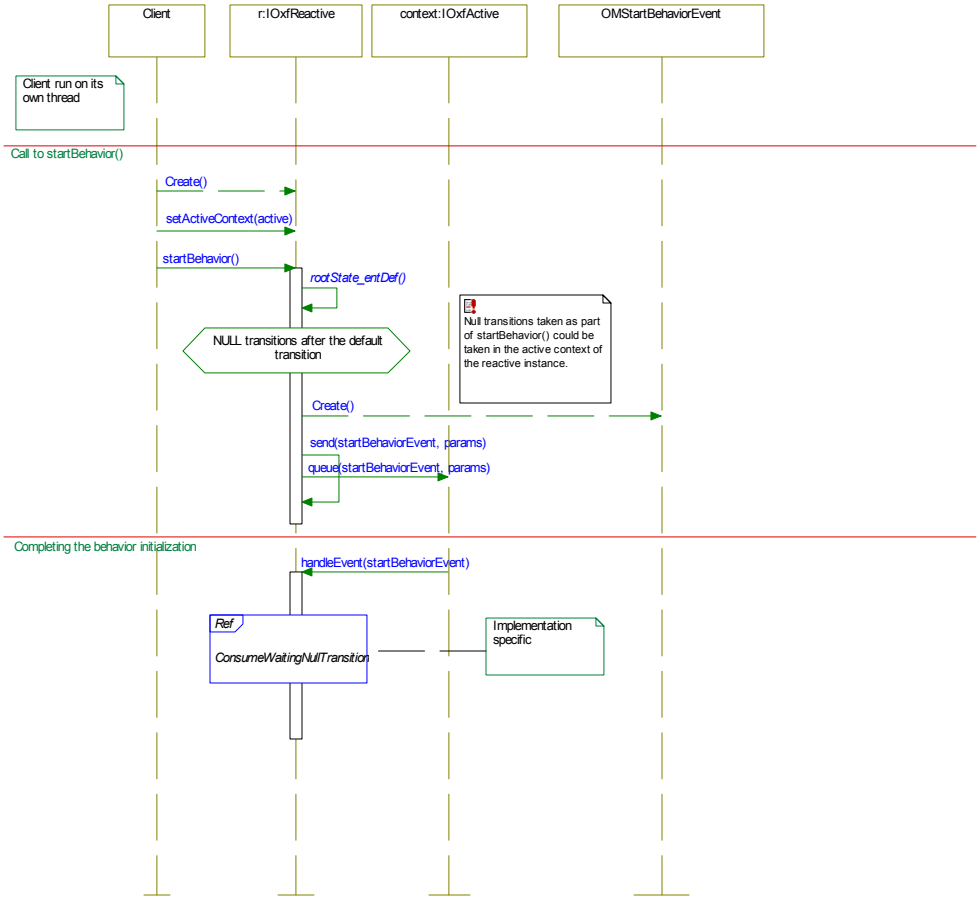
NULL transitions are transitions without triggers.
The UML specifies that a NULL transition should be consumed whenever possible. The place to check where a NULL transition can be consumed is at the end of an event consumption.

This scenario should be addressed by the implementation of IOxfReactive.



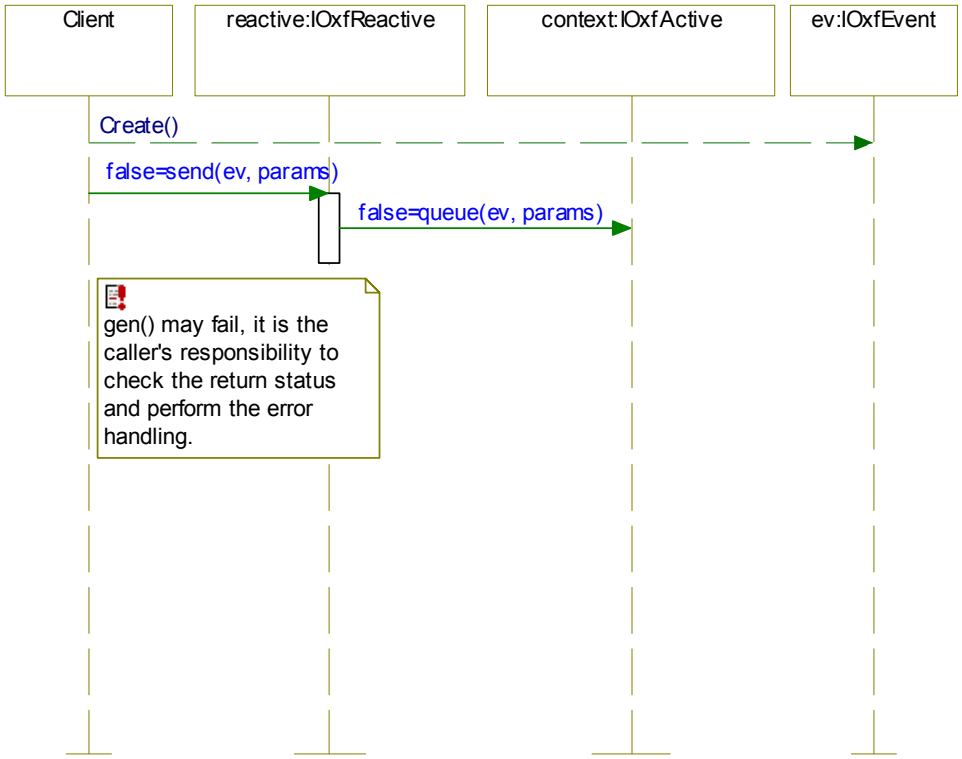
Sequence Diagram name: Initializing statechart behavior

Description: This diagram shows the way a reactive behavior is started.



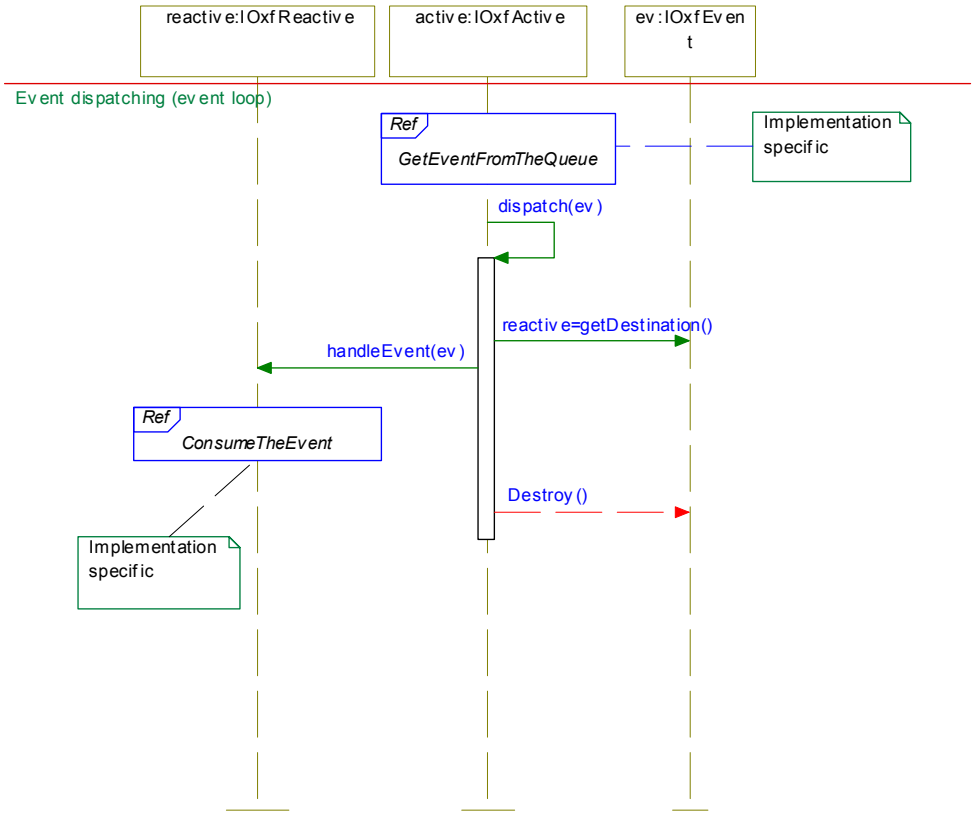
Sequence Diagram name: Generate event fail

Description: Generate event fail



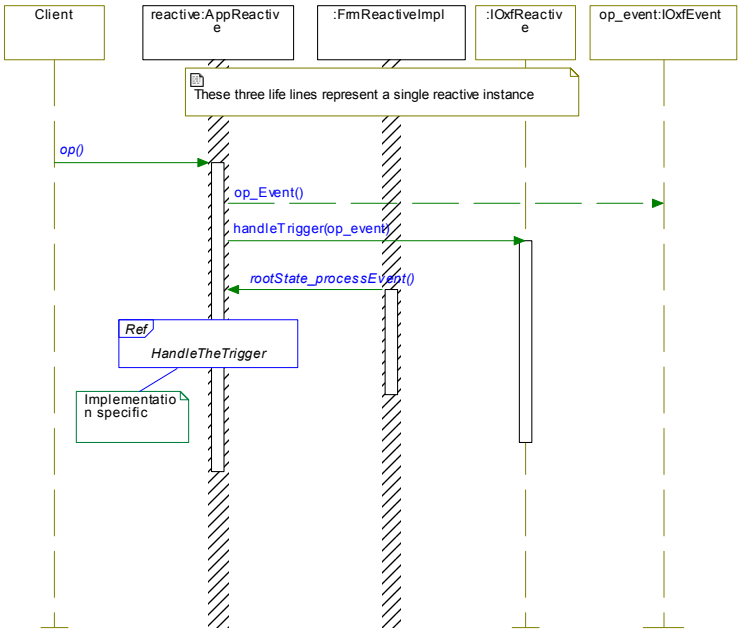
Sequence Diagram name: Event Handling

Description: Event dispatching and processing



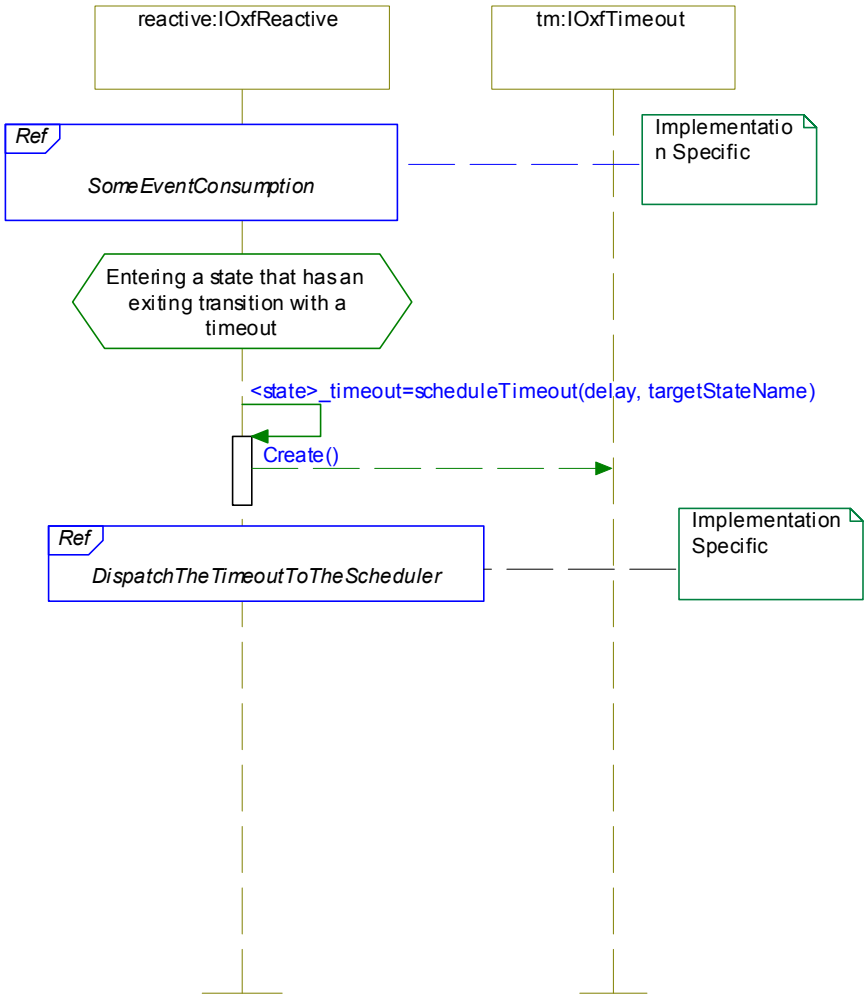
Sequence Diagram name: Handling a triggered operation

Description: Triggered operation call (synchronous event handling)



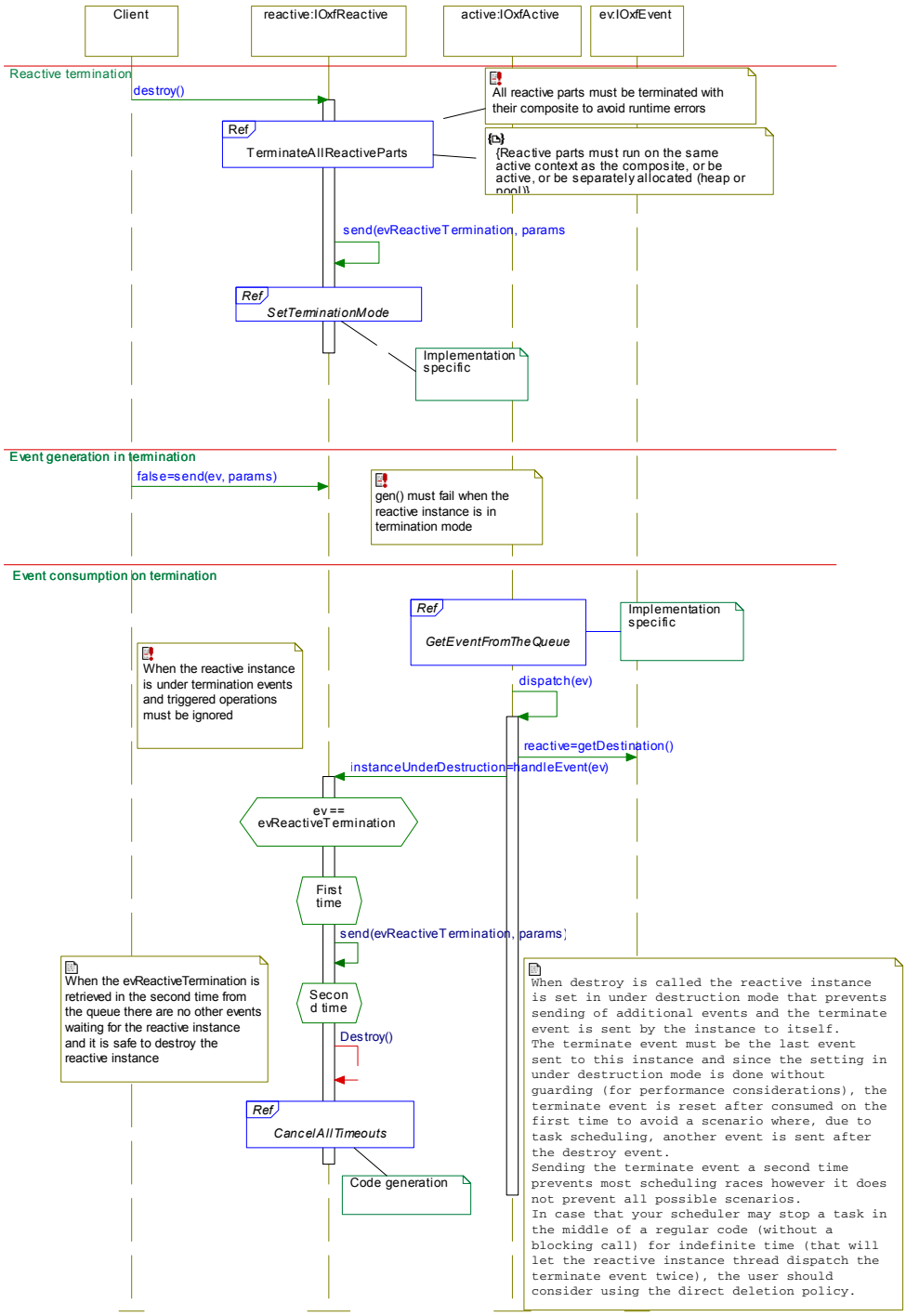
Sequence Diagram name: Scheduling of a timeout

Description: Scheduling of a timeout



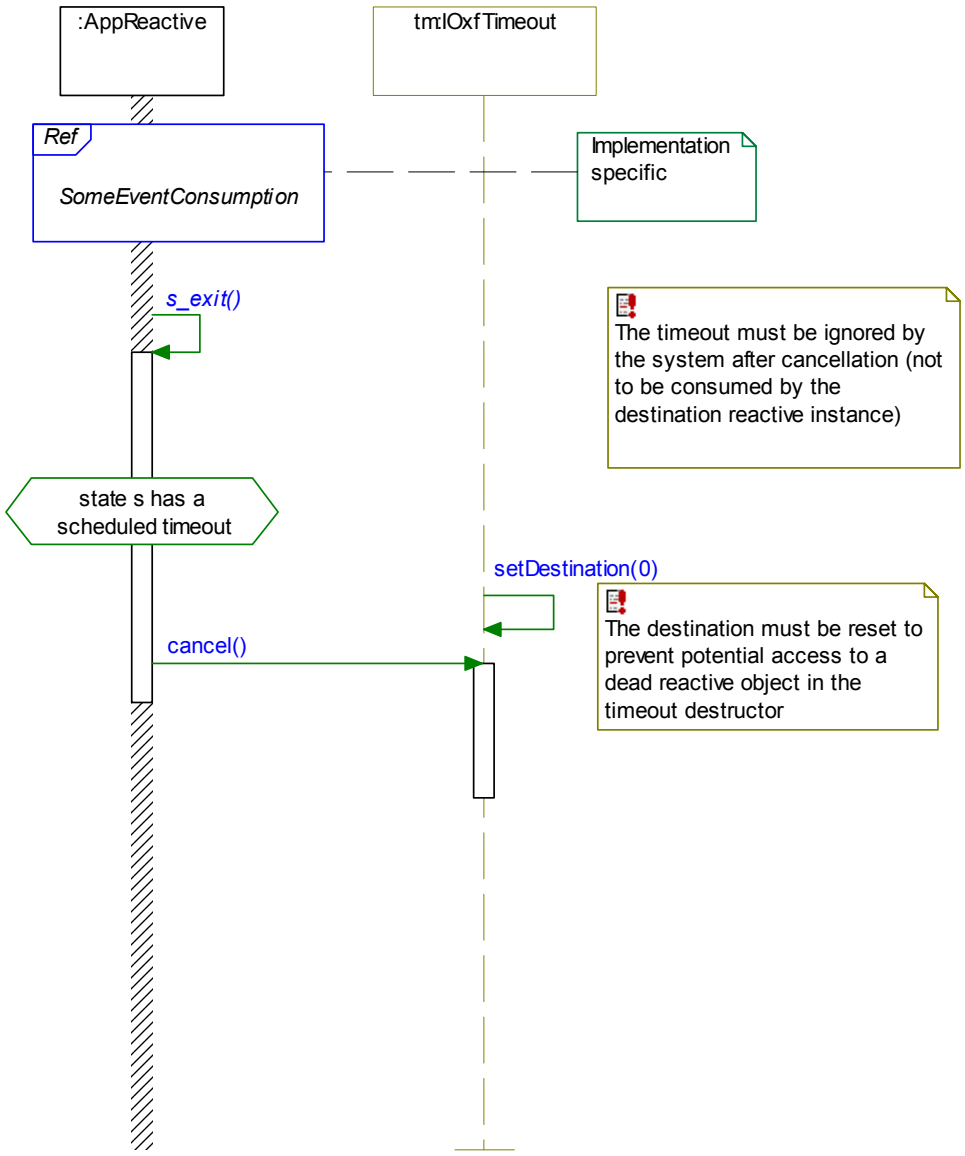
Sequence Diagram name: Termination of a reactive instance

Description: Termination of a reactive instance



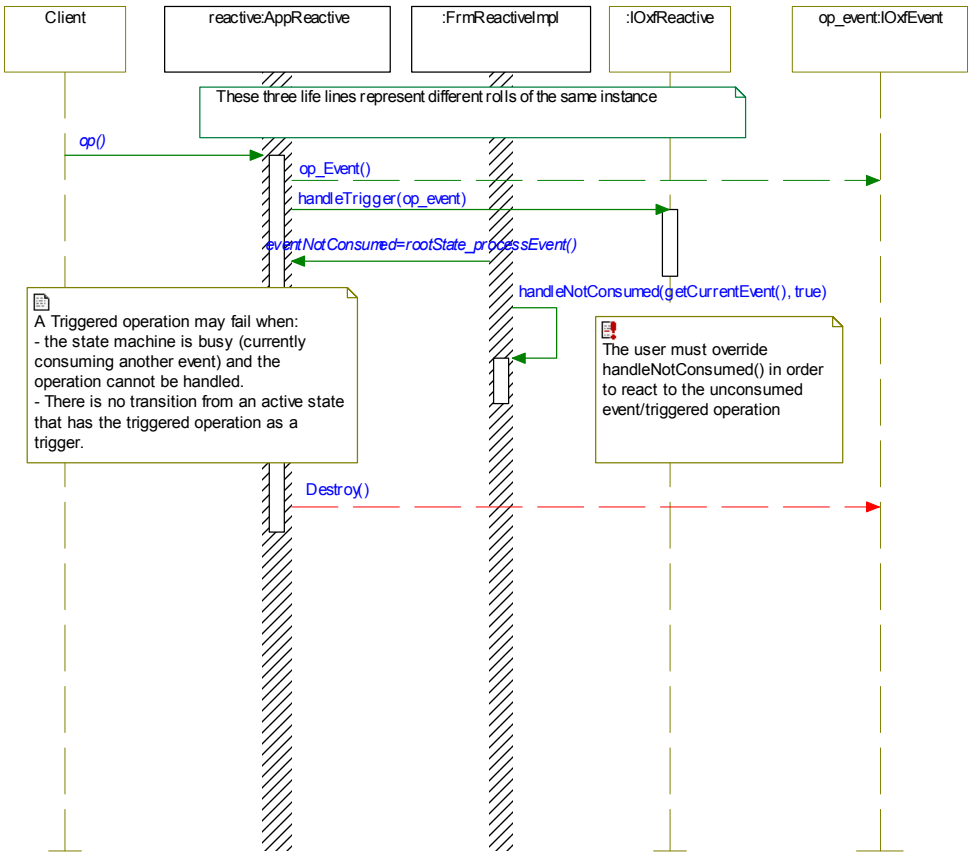
Sequence Diagram name: Cancellation of a scheduled timeout

Description: Cancellation of a scheduled timeout



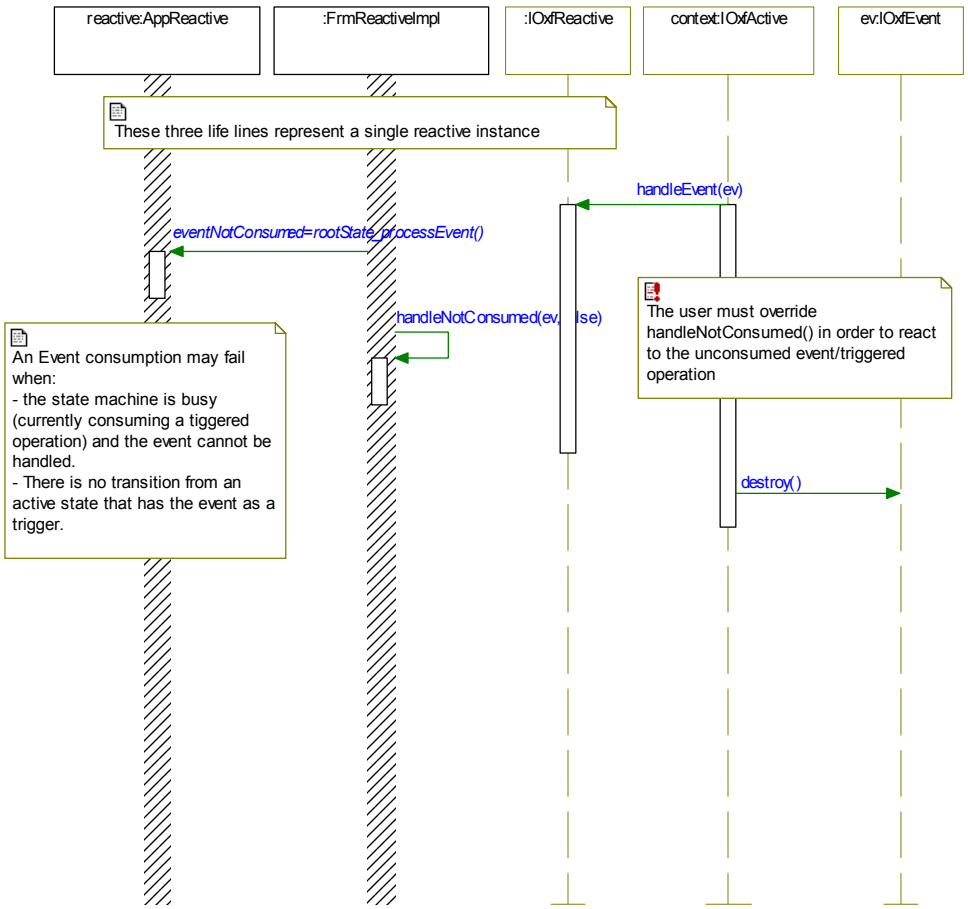
Sequence Diagram name: Handling missed triggers

Description: This diagram shows how the user can identify and handle missed triggered operation calls



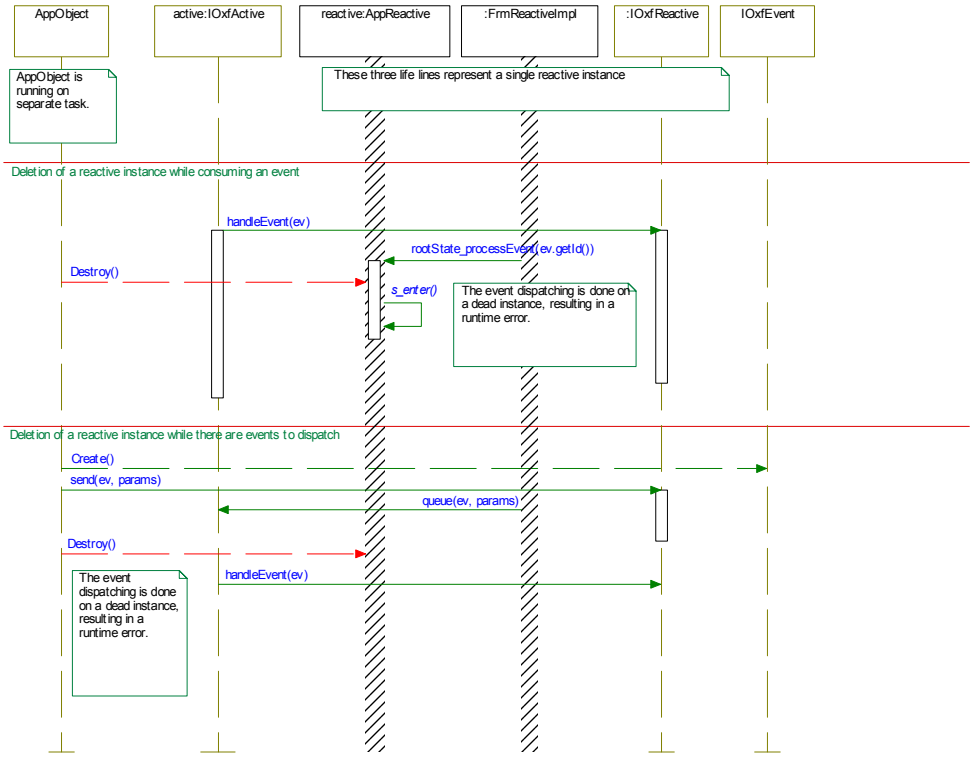
Sequence Diagram name: Handling missed events

Description: This diagram shows how the user can identify and handle missed events



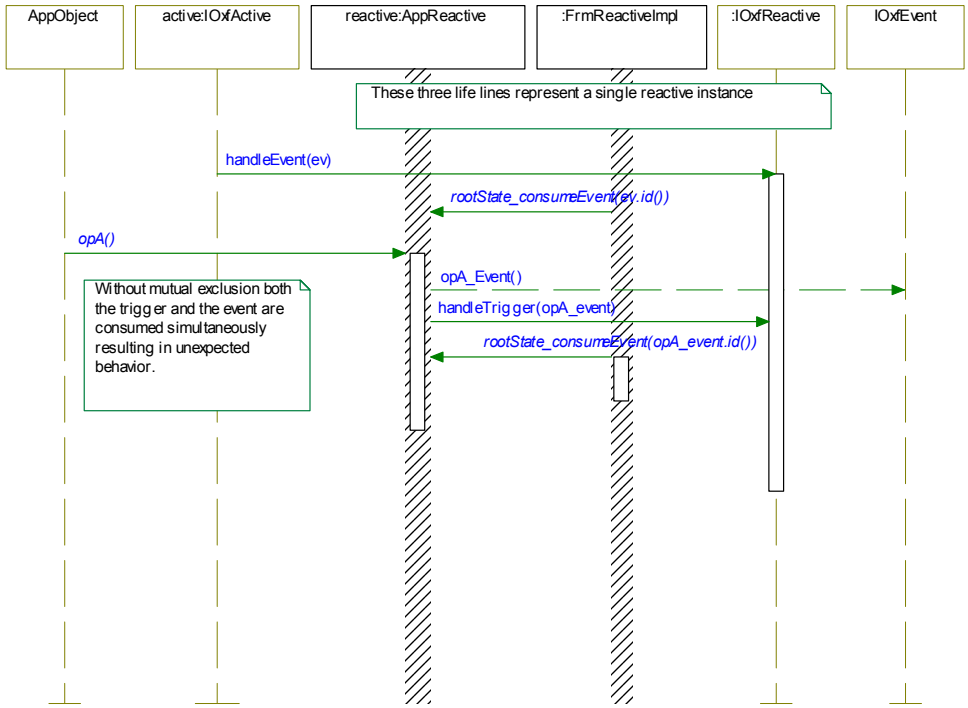
Sequence Diagram name: Race scenarios on reactive instance deletion

Description: Potential race scenarios on reactive instance deletion



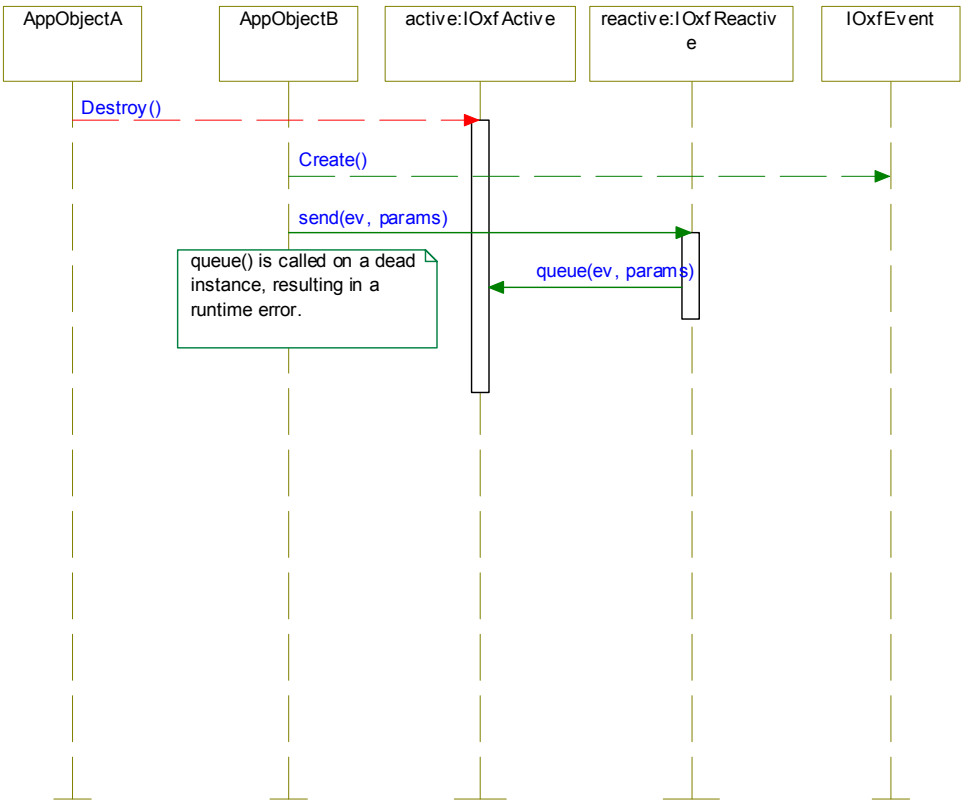
Sequence Diagram name: Race of triggered operation and event consumption

Description: Potential race between event consumption and triggered operation call.



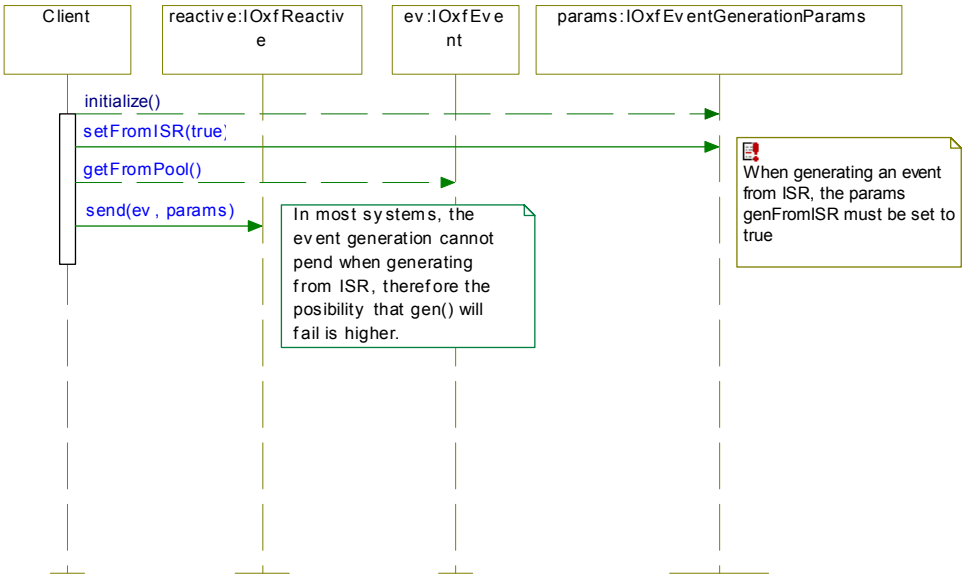
Sequence Diagram name: Race on active instance deletion

Description: Potential race on active instance deletion



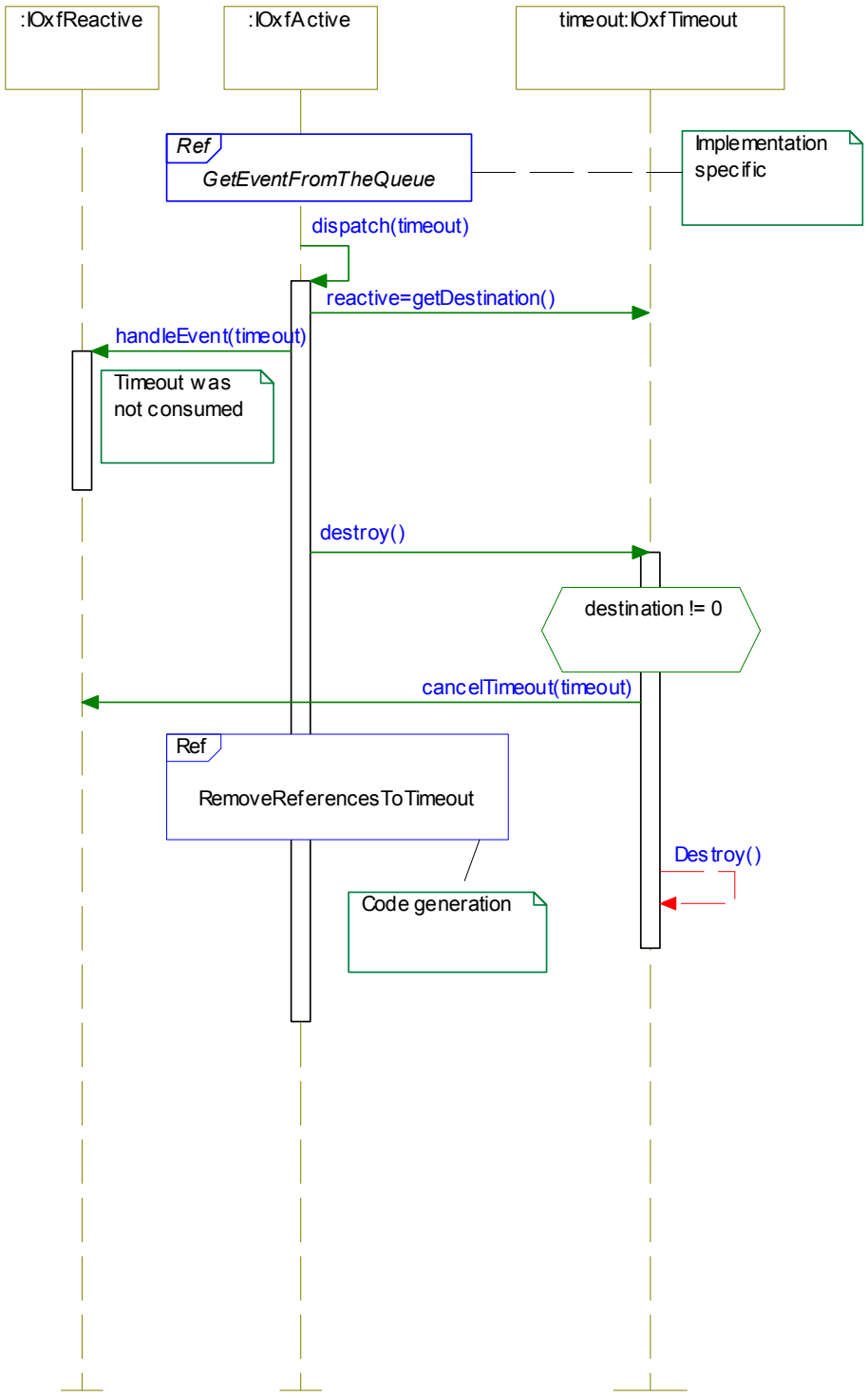
Sequence Diagram name: generation of event from ISR

Description: This is a generation scenario from an ISR call



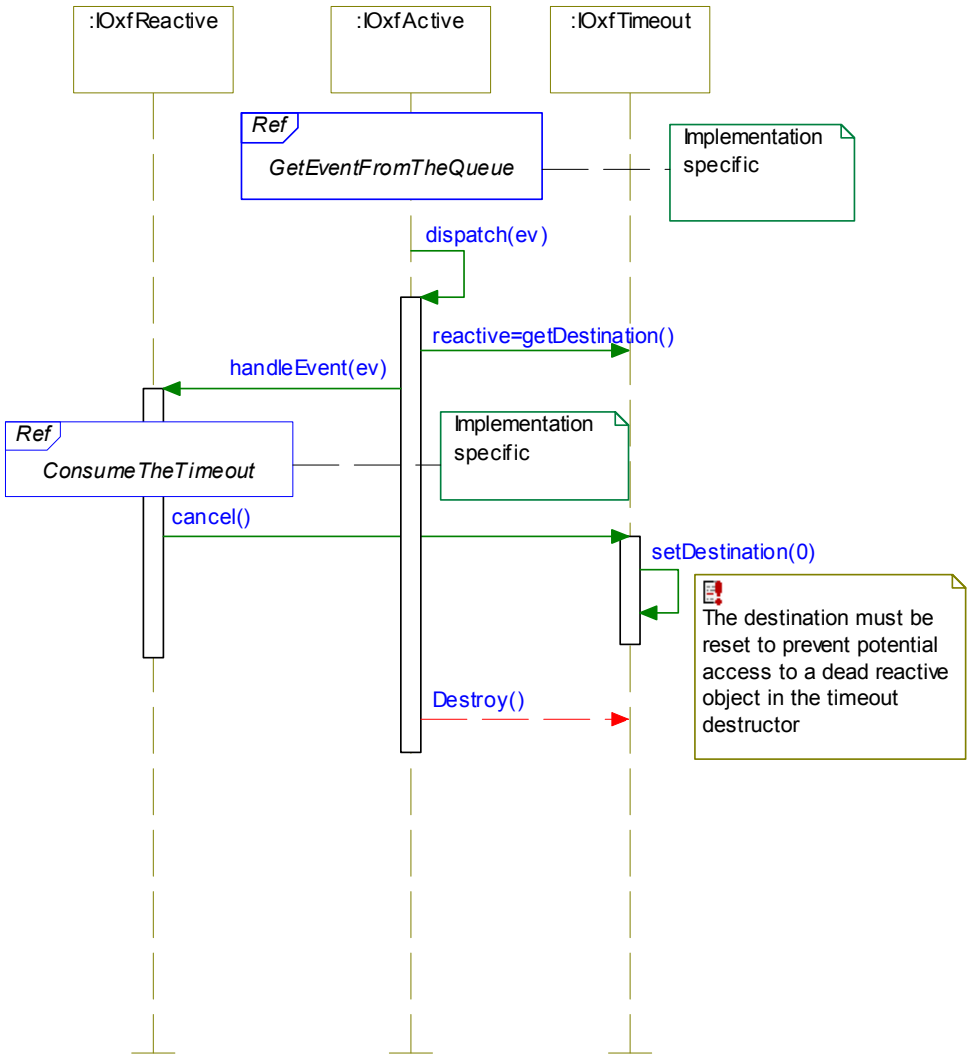
Sequence Diagram name: Unconsumed timeout self cancellation

Description: A timeout may be discarded by a state machine in several scenarios (for example if the timeout target state is no longer active).
This diagram shows the collaboration between the reactive implementation and the timeout that guarantees that the reactive instance will not keep references to dead timeouts.



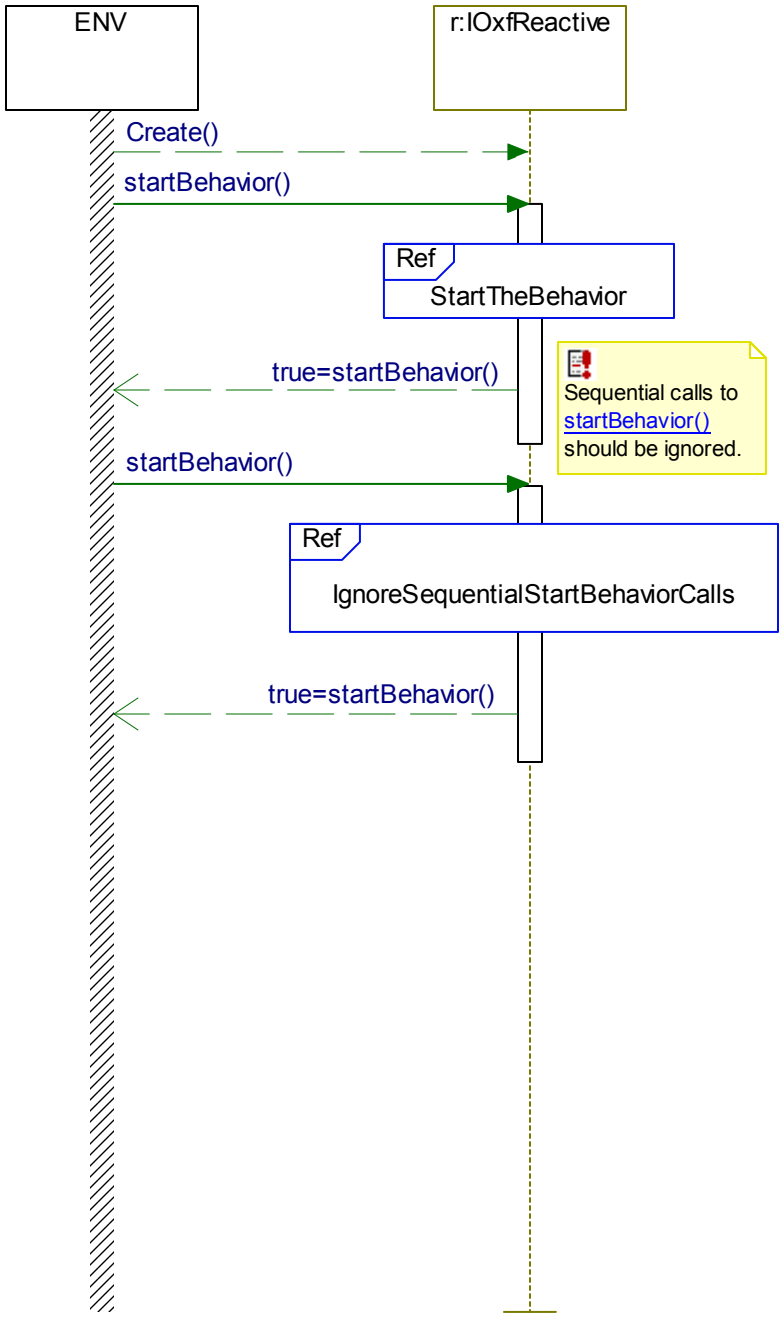
Sequence Diagram name: Timeout consumption

Description: Cancellation of a scheduled timeout



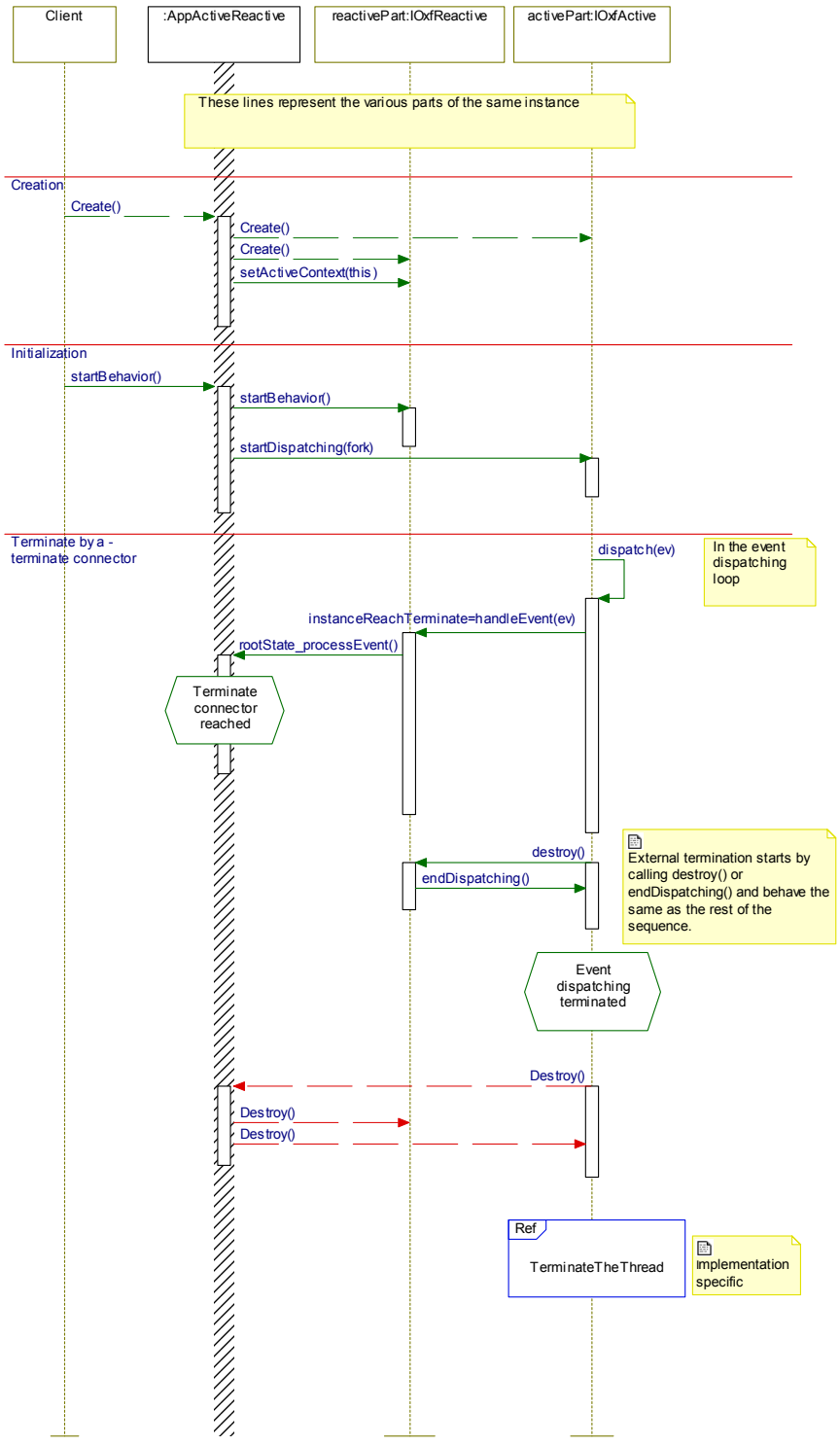
Sequence Diagram name: Sequential calls to startBehavior

Description: This scenario illustrates the behavior when a two or more calls to IOxfReactive::startBehavior() are made.



Sequence Diagram name: Active reactive instance lifecycle

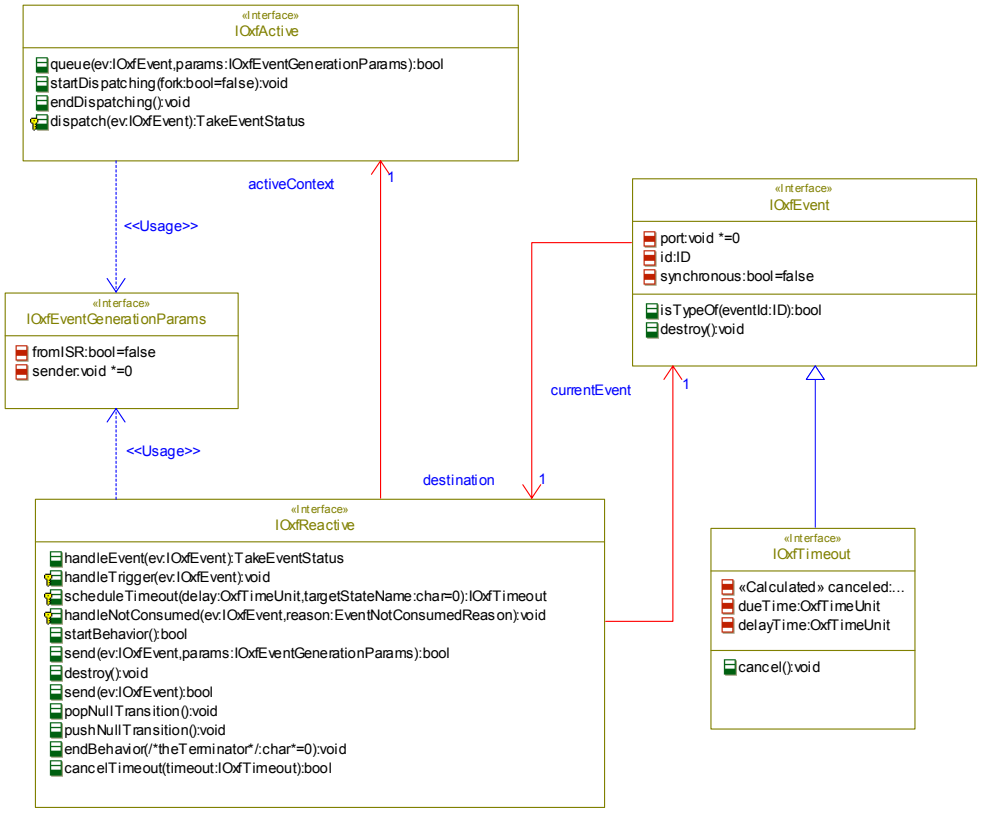
Description:



Object Model Diagram Information

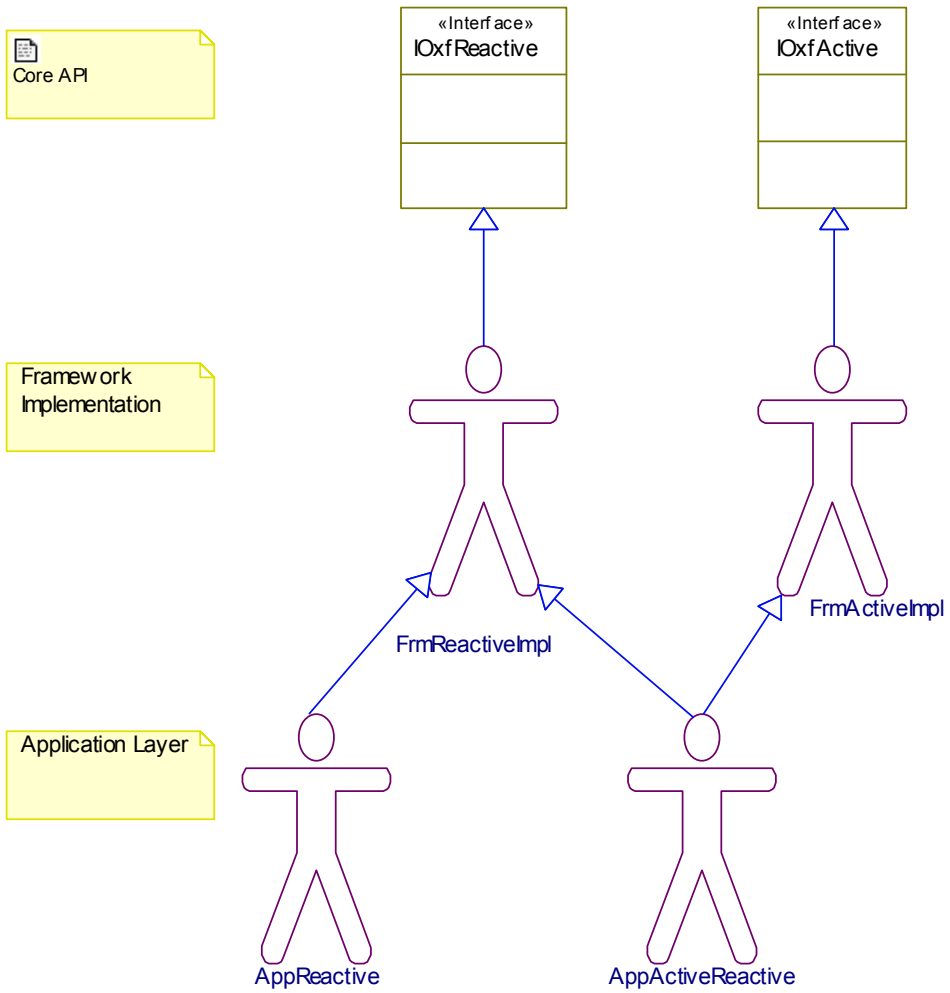
Object Model Diagram name: core interfaces

Description: The relations between the core interface of the OXF



Object Model Diagram name: Actors

Description: Actors used by the CoreAPI sequence diagrams



Actor Information for Package: CoreAPI

Actor name: AppReactive

Description:Application reactive class

Generalization information for Actor AppReactive

Generalization name: FrmReactiveImpl

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
FrmReactiveImpl	FrmReactiveImpl	AppReactive

Actor name: FrmReactiveImpl

Description:The framework reactive Implementation

Operation information for Actor: FrmReactiveImpl

Operation name: rootState_processEvent

Initializer:
Const: false
Trigger: false
Body:
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: rootState_processEvent()
Return Type: void
Description:

Generalization information for Actor FrmReactiveImpl

Generalization name: IOxfReactive

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfReactive	IOxfReactive	FrmReactiveImpl

Actor name: AppActiveReactive

Description:Application active-reactive class

Generalization information for Actor AppActiveReactive

Generalization name: FrmReactiveImpl

Description:
Virtual: false
Visibility: public
Extension Point:

Generalization name: FrmActiveImpl

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
FrmReactiveImpl	FrmReactiveImpl	AppActiveReactive
FrmActiveImpl	FrmActiveImpl	AppActiveReactive

Actor name: FrmActiveImpl

Description:A representation of an implementation of IOxfActive

Generalization information for Actor FrmActiveImpl

Generalization name: IOxfActive

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfActive	IOxfActive	FrmActiveImpl

Class Information for Package: CoreAPI

Class name: IOxfReactive

Description:UML reactive class interface.
Responsible for event consumption.
A reactive instance is consuming events in a context of a single active instance.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: IOxfReactive

Operation name: handleEvent

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: handleEvent(IOxfEvent ev)
Return Type: TakeEventStatus
Description: consume an event

Argument information for Operation handleEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: handleTrigger

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: protected
Signature: handleTrigger(IOxfEvent ev)
Return Type: void
Description: consume a triggered operation (synchronous event)

Argument information for Operation handleTrigger

Name	Type	Direction
ev	IOxfEvent	In

Operation name: scheduleTimeout

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: protected
Signature: scheduleTimeout(OxfTimeUnit delay,char targetStateName)
Return Type: IOxfTimeout
Description: schedule a timeout to be consumed by the reactive instance.

Argument information for Operation scheduleTimeout

Name	Type	Direction
delay	OxfTimeUnit	In
targetStateName	char	In

Operation name: handleNotConsumed

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: protected
Signature: handleNotConsumed(IOxfEvent ev,EventNotConsumedReason reason)
Return Type: void
Description: react to an event that was not consumed.
note that the event can be allocated on the stack.

Argument information for Operation handleNotConsumed

Name	Type	Direction
ev	IOxfEvent	In
reason	EventNotConsumedReason	In

Operation name: startBehavior

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: startBehavior()
Return Type: bool
Description: initialize the reactive instance state machine

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: send(IOxfEvent ev,IOxfEventGenerationParams params)
Return Type: bool
Description: send the specified event to the instance active context queue

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In
params	IOxfEventGenerationParams	In

Operation name: destroy

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: destroy()
Return Type: void
Description: destroy the reactive instance (delete should never be called directly)

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: send(IOxfEvent ev)
Return Type: bool
Description: send the specified event to the instance active context queue

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In

Operation name: ~IOxfReactive

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true

Visibility: public
Signature: ~IOxfReactive()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: popNullTransition

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: popNullTransition()
Return Type: void
Description: signal that a null transition was taken (called by the generated code)

Operation name: pushNullTransition

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: pushNullTransition()
Return Type: void
Description: signal that there is a null transition to be taken (called by the generated code)

Operation name: endBehavior

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: true
Visibility: public
Signature: endBehavior(char* /*theTerminator*/)
Return Type: void
Description: signal that the reactive instance reached a terminate connector

Argument information for Operation endBehavior

Name	Type	Direction
/*theTerminator*/	char*	In

Operation name: cancelTimeout

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public

Signature: cancelTimeout(IOxfTimeout timeout)
Return Type: bool
Description: Cleanup references to the specified timeout.
Return true if the reference was removed.

Argument information for Operation cancelTimeout

Name	Type	Direction
timeout	IOxfTimeout	In

Type information for Class IOxfReactive

Type name: TakeEventStatus

Description: Event dispatching result
Kind: Enumeration

EnumerationLiteral information for Type TakeEventStatus

Name	Value
eventNotConsumed	0
eventConsumed	1
instanceUnderDestruction	2
instanceReachTerminate	3

Type name: EventNotConsumedReason

Description: The reason handleNotConsumed() is called.

StateMachineBusy - the state machine is handling another event (for example when a triggered operation is called while an event is being consumed).
EventNotHandledByStatechart - the event was processed by the state machine but the statechart was not in a state that consumed it.
Kind: Enumeration

EnumerationLiteral information for Type EventNotConsumedReason

Name	Value
StateMachineBusy	
EventNotHandledByStatechart	

Relation information for Class IOxfReactive

Relation name: activeContext

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: activeContext
LinkName:
RoleName: activeContext
Type: Association
Description: The active context of the reactive instance.

Relation name: currentEvent

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: currentEvent
LinkName:
RoleName: currentEvent
Type: Association
Description: The event currently consumed by the reactive instance

Name	Inverse	Source	Target
activeContext		IOxfReactive	IOxfActive
currentEvent		IOxfReactive	IOxfEvent

Class name: IOxfEvent

Description:An event (signal/message) interface.
Events are used for synchronous and asynchronous messaging.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: IOxfEvent

Attribute Name: port

Default Value: 0
Static: false
Visibility: public
Type: void *
Stereotype:
Description: The port that the event was received on.

Attribute Name: id

Default Value:
Static: false
Visibility: public
Type: ID
Stereotype:
Description: The event id.

Attribute Name: synchronous

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: Mark the event as a synchronous event (i.e. triggered operation).

Operation information for Class: IOxfEvent

Operation name: isTypeOf

Initializer:

Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: isTypeOf(ID eventId)
Return Type: bool
Description: check if the event is a sub-type of an event with the specified id

Argument information for Operation isTypeOf

Name	Type	Direction
eventId	ID	In

Operation name: destroy

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: destroy()
Return Type: void
Description: destroy the event

Operation name: ~IOxfEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfEvent()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Type information for Class IOxfEvent

Type name: ID

Description: An event id attribute type
Kind: Typedef
Basic Type: short
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Relation information for Class IOxfEvent

Relation name: destination

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: destination
LinkName:
RoleName: destination
Type: Association
Description: The event destination (an IOxfReactive instance).

Name	Inverse	Source	Target
destination		IOxfEvent	IOxfReactive

Class name: IOxfTimeout

Description:A timeout interface.
A timeout is a unique type of event.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: IOxfTimeout

Attribute Name: canceled

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: When the event is canceled, it should be ignored by the event dispatcher.

Attribute Name: dueTime

Default Value:
Static: false
Visibility: public
Type: OxfTimeUnit
Stereotype:
Description: The absolute time (i.e. system time) until the timeout will expire.
This time is calculated by:
dueTime = <timeout scheduling time> + delayTime.

Attribute Name: delayTime

Default Value:
Static: false
Visibility: public
Type: OxfTimeUnit
Stereotype:
Description: The relative delay until the timeout should be expired.

Operation information for Class: IOxfTimeout

Operation name: ~IOxfTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfTimeout()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: cancel

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: cancel()
Return Type: void
Description: cancel the timeout

Generalization information for Class IOxfTimeout

Generalization name: IOxfEvent

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfEvent	IOxfEvent	IOxfTimeout

Class name: IOxfEventGenerationParams

Description:Abstract event generation parameters (allows the implementer to add generation parameters, such as generation timeout, priority etc.)
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: IOxfEventGenerationParams

Attribute Name: fromISR

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:

Description: Should be set to true when the event is generated by an ISR

Attribute Name: sender

Default Value: 0
Static: false
Visibility: public
Type: void *
Stereotype:
Description: The event sender (used for instrumentation).

Operation information for Class: IOxfEventGenerationParams

Operation name: ~IOxfEventGenerationParams

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfEventGenerationParams()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Class name: IOxfActive

Description:Active class interface.
An active class is an event dispatcher executing an event loop (message pump), potentially on its own thread of execution.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: IOxfActive

Operation name: queue

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: queue(IOxfEvent ev,IOxfEventGenerationParams params)
Return Type: bool
Description: Queue the specified event to be handled by the event loop asynchronously.

Argument information for Operation queue

Name	Type	Direction
ev	IOxfEvent	In
params	IOxfEventGenerationParams	In

Operation name: startDispatching

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: startDispatching(bool fork)
Return Type: void
Description: start the event loop

Argument information for Operation startDispatching

Name	Type	Direction
fork	bool	In

Operation name: endDispatching

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: endDispatching()
Return Type: void
Description: end the active instance event dispatching and destroy the instance.

Operation name: dispatch

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: protected
Signature: dispatch(IOxfEvent ev)
Return Type: TakeEventStatus
Description: dispatch the specified event to its destination

Argument information for Operation dispatch

Name	Type	Direction
ev	IOxfEvent	In

Operation name: ~IOxfActive

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public

Signature: ~IOxfActive()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Package Information

Description: Interfaces exposing the framework Core API

Package: Types

Type information for Package Types

Type name: OxfTimeUnit

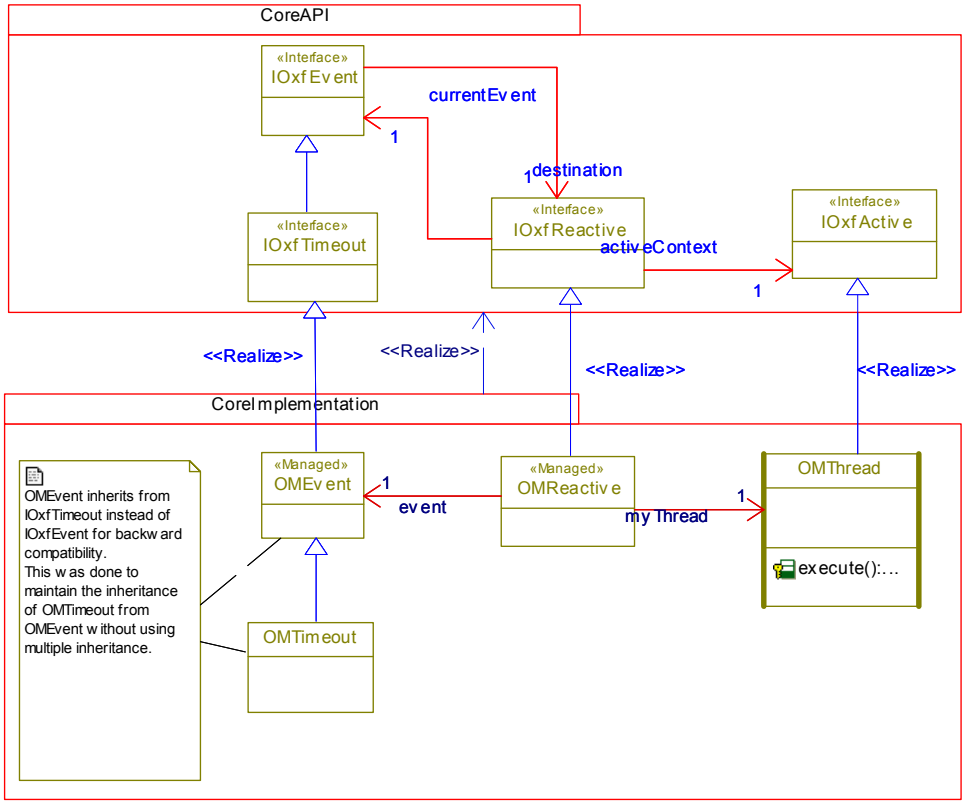
Description: Time measurement
Kind: Typedef
Basic Type: unsigned long
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Package: CoreImplementation

Object Model Diagram Information

Object Model Diagram name: Core API implementation

Description: This diagram shows the implementation of the core API



Class Information for Package: CoreImplementation

Class name: OMReactive

Description:The base IOxfReactive implementation
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMReactive

Attribute Name: defaultStateMask

Default Value: 0x00000000L
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - the initial state

Attribute Name: nullTransitionStateMask

Default Value: 0x00000001L
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - a null transition was found/taken

Attribute Name: nullTransitionMask

Default Value: 0x0000FFFFL
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - the max number of waiting null transitions that OMReactive can handle

Attribute Name: terminateConnectorReachedStateMask

Default Value: 0x00010000L
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - the instance statechart reached a terminate connector

Attribute Name: underDestructionStateMask

Default Value: 0x00020000L
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - the instance is under destruction

Attribute Name: deleteOnTerminateStateMask

Default Value: 0x00040000L
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - the reactive instance should be deleted when a terminate connector is reached

Attribute Name: shouldCompleteStartBehaviorStateMask

Default Value: 0x00080000L
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - need to complete the startBehavior() by a start behavior event

Attribute Name: behaviorStartedStateMask

Default Value: 0x00100000L
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - startBehavior() was called

Attribute Name: maxNullSteps

Default Value: DEFAULT_MAX_NULL_STEPS
Static: true
Visibility: public
Type: int
Stereotype:
Description: Maximum number of null transitions allowed in a single take
infinite loop checker.

Attribute Name: state

Default Value: 0
Static: false
Visibility: protected
Type: unsigned long
Stereotype:
Description: The reactive internal state (bit field)

Attribute Name: frameworkInstance

Default Value: false
Static: false
Visibility: protected
Type: bool
Stereotype:
Description: This flag mark an instance as an internal to the framework (that should be invisible in design-level debugging)

Attribute Name: active

Default Value: false
Static: false

Visibility: public
Type: bool
Stereotype:
Description: This flag mark an active-reactive class

Attribute Name: busy

Default Value: false
Static: false
Visibility: protected
Type: bool
Stereotype:
Description: This flag indicate that the reactive instance is currently handling an event.

Attribute Name: toGuardReactive

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: This flag indicate that the destruction of the reactive instance should be guarded against mutual exclusion - prevents deletion of the instance while processing an event.

Attribute Name: theStartBehaviorEvent

Declaration: OMStartBehaviorEvent
Default Value:
Static: false
Visibility: private
Type:
Stereotype:
Description: The instance start behavior event.
This event is sent from the instance to itself when the default transition is followed by null transitions.

Attribute Name: OMTakeEventCompletedEventNotConsumed

Default Value: eventNotConsumed
Static: true
Visibility: public
Type: TakeEventStatus
Stereotype:
Description: Alias to IOxfReactive::eventNotConsumed

Attribute Name: OMTakeEventCompleted

Default Value: eventConsumed
Static: true
Visibility: public
Type: TakeEventStatus
Stereotype:
Description: Alias to IOxfReactive::eventConsumed

Attribute Name: supportDirectDeletion

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:

Description: When set to true, the reactive class should support direct deletion.
The user can also set the globalSupportDirectDeletion attribute to affect all the reactive classes in the system.
The reactive instance will support direct deletion if either of the attributes is set to true.

Attribute Name: globalSupportDirectDeletion

Default Value: false
Static: true
Visibility: public
Type: bool
Stereotype:
Description: When set to true, all the reactive classes in the system should support direct deletion.
The user can also set the supportDirectDeletion attribute to affect a single reactive instance.
The reactive instance will support direct deletion if either of the attributes is set to true.

Attribute Name: theTerminateEvent

Declaration: OMReactiveTerminationEvent
Default Value:
Static: false
Visibility: private
Type:
Stereotype:
Description: The reactive terminate event.
The reactive instance sends this event to itself on call to terminate() and will self-destruct once the event is consumed

Attribute Name: destroyEventResentStateMask

Default Value: 0x00200000L
Static: true
Visibility: protected
Type: unsigned long
Stereotype:
Description: OMReactive state mask - indicate that theTerminateEvent was resent.
The terminate event is send twice to reduce the probability of a race between event sending on call to destroy().
When destroy is called the reactive instance is set in under destruction mode that prevents sending of additional events and the terminate event is sent by the instance to itself.
The terminate event must be the last event sent to this instance and since the setting in under destruction mode is done without guarding (for performance considerations), the terminate event is reset after consumed on the first time to avoid a scenario where, due to task scheduling, another event is sent after the destroy event.
Sending the terminate event a second time prevents most scheduling races however it does not prevent all possible scenarios.
In case that your scheduler may stop a task in the middle of a regular code (without a blocking call) for indefinite time (that will let the reactive instance thread dispatch the terminate event twice), you should consider using the direct deletion policy.

Operation information for Class: OMReactive

Operation name: ~OMReactive

Initializer:
Const: false
Trigger: false
Abstract: false

Static: false
Virtual: true
Visibility: public
Signature: ~OMReactive()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: OMReactive

Initializer: event(0), myThread(0), rootState(0), activeContext(0), currentEvent(0), eventGuard(0)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMReactive(IOxfActive context)
Return Type:
Description: Initialize the reactive instance

Argument information for Operation OMReactive

Name	Type	Direction
context	IOxfActive	In

Operation name: handleEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: handleEvent(IOxfEvent ev)
Return Type: TakeEventStatus
Description: Consume an event

Argument information for Operation handleEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: send(IOxfEvent ev,IOxfEventGenerationParams params)
Return Type: bool
Description: send an event to the active context queue

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In
params	IOxfEventGenerationParams	In

Operation name: startBehavior

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: startBehavior()
Return Type: bool
Description: initialize the reactive instance state machine

Operation name: destroy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: destroy()
Return Type: void
Description: destroy the reactive instance (delete should never be called directly)

Operation name: setActiveContext

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: setActiveContext(IOxfActive context)
Return Type: void
Description: set the active context

Argument information for Operation setActiveContext

Name	Type	Direction
context	IOxfActive	In

Operation name: setCurrentEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false

Visibility: protected
Signature: setCurrentEvent(IOxfEvent ev)
Return Type: void
Description: set the current event

Argument information for Operation setCurrentEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: scheduleTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: scheduleTimeout(OxfTimeUnit delay,char targetStateName)
Return Type: IOxfTimeout
Description: schedule a timeout to be consumed by the reactive instance.

Argument information for Operation scheduleTimeout

Name	Type	Direction
delay	OxfTimeUnit	In
targetStateName	char	In

Operation name: processEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: processEvent(IOxfEvent ev)
Return Type: TakeEventStatus
Description: events/triggered operations event consumption shared code

Argument information for Operation processEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: rootState_entDef

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: rootState_entDef()

Return Type: void
Description: Take the statechart initial default transition(s).
In FLAT statechart code, expected to be overridden by the user class.

Operation name: rootState_processEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: rootState_processEvent()
Return Type: TakeEventStatus
Description: Dispatch the received event to the statechart.
In FLAT statechart code, expected to be overridden by the user class.

Operation name: handleEventNotConsumed

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: handleEventNotConsumed(OMEvent /*ev*/)
Return Type: void
Description: Deprecated.
handle unconsumed event.
Use handleNotConsumed()

Argument information for Operation handleEventNotConsumed

Name	Type	Direction
/*ev*/	OMEvent	In

Operation name: handleTONotConsumed

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: handleTONotConsumed(OMEvent /*ev*/)
Return Type: void
Description: Deprecated.
handle unconsumed event.
Use handleNotConsumed()

Argument information for Operation handleTONotConsumed

Name	Type	Direction
/*ev*/	OMEvent	In

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: send(IOxfEvent ev)
Return Type: bool
Description: send the specified event to the instance active context queue

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In

Operation name: cancelEvents

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cancelEvents()
Return Type: void
Description: cleanup the event queue in destruction,
if the user modified the event queue - reactive relationship,
it must deal with the cleanup in its own code.

Operation name: hasWaitingNullTransitions

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: hasWaitingNullTransitions()
Return Type: bool
Description: check if there are null transitions to take

Operation name: isBehaviorStarted

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: isBehaviorStarted()
Return Type: bool
Description: getter/setter for the behavior started flag

Operation name: IsCurrentEvent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: IsCurrentEvent(ID eventId)
Return Type: bool
Description: Check if the provided event ID matches the current handled event.

Argument information for Operation IsCurrentEvent

Name	Type	Direction
eventId	ID	In

Operation name: popNullTransition

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: popNullTransition()
Return Type: void
Description: signal that a null transition was taken (called by the generated code)

Operation name: pushNullTransition

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: pushNullTransition()
Return Type: void
Description: signal that there is a null transition to be taken (called by the generated code)

Operation name: runToCompletion

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: runToCompletion()
Return Type: void
Description: take null transitions - called at the end of the event consumption

Operation name: setCompleteStartBehavior

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: setCompleteStartBehavior(bool b)
Return Type: void
Description: set the ShouldCompleteStartBehavior flag

Argument information for Operation setCompleteStartBehavior

Name	Type	Direction
b	bool	In

Operation name: setUnderDestruction

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: setUnderDestruction()
Return Type: void
Description: Mark that the instance is being destroyed

Operation name: setShouldDelete

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setShouldDelete(bool b)
Return Type: void
Description: Mark the instance as dynamically or statically allocated

Argument information for Operation setShouldDelete

Name	Type	Direction
b	bool	In

Operation name: setShouldTerminate

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public

Signature: setShouldTerminate(bool b)
Return Type: void
Description: getter/setter for the shouldTerminate flag

Argument information for Operation setShouldTerminate

Name	Type	Direction
b	bool	In

Operation name: setActiveContext

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: setActiveContext(IOxfActive context,bool activeInstance)
Return Type: void
Description: this operation is virtual and public in order to allow user to initialize
nested embedded reactive components of active class manually
Rhapsody supports only one level of nesting for such cases.
The usage of the operation should be done with care.

Argument information for Operation setActiveContext

Name	Type	Direction
context	IOxfActive	In
activeInstance	bool	In

Operation name: setToGuardReactive

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setToGuardReactive(bool flag)
Return Type: void
Description: mark the reactive instance as guarded - to prevent mutual exclusion between the instance deletion and the event consumption

Argument information for Operation setToGuardReactive

Name	Type	Direction
flag	bool	In

Operation name: shouldCompleteRun

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: protected
Signature: shouldCompleteRun()
Return Type: bool
Description: check if there are null transitions to take

Operation name: shouldCompleteStartBehavior

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: shouldCompleteStartBehavior()
Return Type: bool
Description: check if there are null transitions to take as part of startBehavior()

Operation name: shouldDelete

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: shouldDelete()
Return Type: bool
Description: Check if the instance is dynamically allocated.

Operation name: shouldTerminate

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: shouldTerminate()
Return Type: bool
Description: Test if the reactive instance should terminate

Operation name: handleTrigger

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: handleTrigger(IOxfEvent ev)
Return Type: void
Description: the entry-point for triggered operation consumption

Argument information for Operation handleTrigger

Name	Type	Direction
ev	IOxfEvent	In

Operation name: endBehavior

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: endBehavior(char* aomArg(theTerminator))
Return Type: void
Description: signal that the reactive instance reached a terminate connector

Argument information for Operation endBehavior

Name	Type	Direction
aomArg(theTerminator)	char*	In

Operation name: isUnderDestruction

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: isUnderDestruction()
Return Type: bool
Description: return true when the instance is under destruction

Operation name: setBehaviorStarted

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: setBehaviorStarted()
Return Type: void
Description: Mark that startBehavior() was called

Operation name: popNullConfig

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public

Signature: popNullConfig()
Return Type: void
Description: Deprecated, use popNullTransition()
signal that a null transition was taken (called by the generated code)

Operation name: pushNullConfig

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: pushNullConfig()
Return Type: void
Description: Deprecated, use pushNullTransition()
signal that there is a null transition to be taken (called by the generated code)

Operation name: serializeStates

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: serializeStates(AOMSState aomsState)
Return Type: void
Description: serialization of the active states vector

Argument information for Operation serializeStates

Name	Type	Direction
aomsState	AOMSState	In

Operation name: rootState_serializeStates

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: rootState_serializeStates(AOMSState aomsState)
Return Type: void
Description: states serialization (should be overridden in Flat code generation)

Argument information for Operation rootState_serializeStates

Name	Type	Direction
aomsState	AOMSState	In

Operation name: getThread

Initializer:

Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getThread()
Return Type: OMThread
Description: Deprecated.
get the active context as OMThread* for backward compatibility.
Use getActiveContext()

Operation name: setThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: setThread(OMThread thread,bool activeInstance)
Return Type: void
Description: Deprecated, use setActiveContext().
This operation is virtual and public in order to allow user to initialize nested embedded reactive components of active class manually
.

Rhapsody supports only one level of nesting for such cases.
The usage of the operation should be done with care.

Argument information for Operation setThread

Name	Type	Direction
thread	OMThread	In
activeInstance	bool	In

Operation name: gen

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: gen(OMEvent ev,bool fromISR)
Return Type: bool
Description: send the specified event to the instance active context queue

Argument information for Operation gen

Name	Type	Direction
ev	OMEvent	In
fromISR	bool	In

Operation name: cancelTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cancelTimeout(IOxfTimeout /*timeout*/)
Return Type: bool
Description: cleanup references to the specified timeout

Argument information for Operation cancelTimeout

Name	Type	Direction
/*timeout*/	IOxfTimeout	In

Operation name: handleNotConsumed

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: handleNotConsumed(IOxfEvent /*ev*/,EventNotConsumedReason /*reason*/)
Return Type: void
Description: react to an event that was not consumed.
note that the event can be allocated on the stack.

Argument information for Operation handleNotConsumed

Name	Type	Direction
/*ev*/	IOxfEvent	In
/*reason*/	EventNotConsumedReason	In

Operation name: takeEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: takeEvent(OMEvent ev)
Return Type: TakeEventStatus
Description: Provide a backward compatibility for users that customized the framework in Rhapsody 5.X

Argument information for Operation takeEvent

Name	Type	Direction
ev	OMEvent	In

Operation name: takeTrigger

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: takeTrigger(OMEvent ev)
Return Type: void
Description: Provide a backward compatibility for users that customized the framework in Rhapsody 5.X

Argument information for Operation takeTrigger

Name	Type	Direction
ev	OMEvent	In

Operation name: consumeEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: consumeEvent(OMEvent ev)
Return Type: TakeEventStatus
Description: Process the event

Argument information for Operation consumeEvent

Name	Type	Direction
ev	OMEvent	In

Operation name: gen

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: gen(OMEvent ev,void * sender)
Return Type: bool
Description: send the specified event to the instance active context queue

Argument information for Operation gen

Name	Type	Direction
ev	OMEvent	In
sender	void *	In

Operation name: _gen

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: _gen(OMEvent ev,bool fromISR)
Return Type: bool
Description: send the specified event to the reactive instance's active context queue

Argument information for Operation _gen

Name	Type	Direction
ev	OMEvent	In
fromISR	bool	In

Operation name: _takeTrigger

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: _takeTrigger(OMEvent ev)
Return Type: void
Description: Provide a backward compatibility signature for users that customized the framework

Argument information for Operation _takeTrigger

Name	Type	Direction
ev	OMEvent	In

Operation name: rootState_dispatchEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: rootState_dispatchEvent(ID /*id*/)
Return Type: int
Description: Dispatch the received event to the statechart.
In FLAT statechart code, expected to be overridden by the user class.

Argument information for Operation rootState_dispatchEvent

Name	Type	Direction
/*id*/	ID	In

Operation name: setInDtor

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: setInDtor()
Return Type: void
Description: Mark that the instance is being destroyed

Operation name: setEventGuard

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setEventGuard(OMProtected guard)
Return Type: void
Description: set the event guard by reference

Argument information for Operation setEventGuard

Name	Type	Direction
guard	OMProtected	In

Operation name: sendEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: sendEvent(IOxfEvent ev,IOxfEventGenerationParams params)
Return Type: bool
Description: actually send an event to the active context queue

Argument information for Operation sendEvent

Name	Type	Direction
ev	IOxfEvent	In
params	IOxfEventGenerationParams	In

Operation name: shouldSupportDirectDeletion

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false

Visibility: public
Signature: shouldSupportDirectDeletion()
Return Type: bool
Description: Check if the instance should support direct deletion

Operation name: terminate

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: terminate(char* theTerminator)
Return Type: void
Description: signal that the reactive instance reached a terminate connector

Argument information for Operation terminate

Name	Type	Direction
theTerminator	char*	In

Operation name: incarnateTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: incarnateTimeout(short id,timeUnit delay,OMHandle theState)
Return Type: OMTimeout
Description: Backward compatibility: create a timeout without scheduling it.

Argument information for Operation incarnateTimeout

Name	Type	Direction
id	short	In
delay	timeUnit	In
theState	OMHandle	In

Operation name: discarnateTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: discarnateTimeout(OMTimeout timeout)
Return Type: void
Description: Backward compatibility: destroy a timeout.

Argument information for Operation discarnateTimeout

Name	Type	Direction
timeout	OMTimeout	In

Operation name: setDestroyEventResent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: setDestroyEventResent()
Return Type: void
Description: Mark that the theTerminateEvent was resent

Operation name: isDestroyEventResent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: isDestroyEventResent()
Return Type: bool
Description: Check if the theTerminateEvent was resent

Operation name: handleEventUnderDestruction

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: handleEventUnderDestruction(IOxfEvent ev)
Return Type: TakeEventStatus
Description: Consume an event when the reaction instance is under destruction

Argument information for Operation handleEventUnderDestruction

Name	Type	Direction
ev	IOxfEvent	In

Type information for Class OMReactive

Type name: Defaults

Description: Constant default values
Kind: Enumeration

EnumerationLiteral information for Type Defaults

Name	Value
DEFAULT_MAX_NULL_STEPS	100

Generalization information for Class OMReactive

Generalization name: IOxfReactive

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfReactive	IOxfReactive	OMReactive

Relation information for Class OMReactive

Relation name: activeContext

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: activeContext
LinkName:
RoleName: activeContext
Type: Association
Description:

Relation name: currentEvent

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: currentEvent
LinkName:
RoleName: currentEvent
Type: Association
Description:

Relation name: rootState

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: rootState
LinkName:
RoleName: rootState
Type: Association
Description:

Relation name: eventGuard

Symmetric: false

Multiplicity: 1
Qualifier:
Visibility: public
Label: eventGuard
LinkName:
RoleName: eventGuard
Type: Association
Description:

Relation name: myThread

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: myThread
LinkName:
RoleName: myThread
Type: Association
Description: Direct reference to OMThread for backward compatibility

Relation name: event

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: event
LinkName:
RoleName: event
Type: Association
Description: Backward compatibility API

Name	Inverse	Source	Target
activeContext		OMReactive	IOxfActive
currentEvent		OMReactive	IOxfEvent
rootState		OMReactive	OMComponentState
eventGuard		OMReactive	OMProtected
myThread		OMReactive	OMThread
event		OMReactive	OMEvent

Class name: OMThread

Description:The base IOxfActive implementation
Active: true
Behavior Overridden: false
Composite: true
Reactive: false

Attribute Information for Class: OMThread

Attribute Name: deletionAllowed

Default Value: true
Static: false
Visibility: public
Type: bool
Stereotype:

Description: Indicator that prevents deletion of statically allocated objects.

Attribute Name: toGuardThread

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: activate guard flag

Attribute Name: processing

Default Value: true
Static: false
Visibility: private
Type: bool
Stereotype:
Description: The event loop state, when the value becomes false the event loop is terminated and the thread is destroyed.

Attribute Name: endOfProcess

Default Value: false
Static: true
Visibility: public
Type: bool
Stereotype:
Description: This flag indicate that the application is terminating.

Attribute Name: dispatching

Default Value: true
Static: false
Visibility: private
Type: bool
Stereotype:
Description: The event loop state, when the value becomes false the event loop will ignore all the events in the queue until the OMEndThreadEvent is received.

Attribute Name: endThreadEvent

Declaration: OMEndThreadEvent
Default Value:
Static: false
Visibility: private
Type:
Stereotype:
Description: The thread termination self event

Attribute Name: finalTermination

Default Value: false
Static: false
Visibility: private
Type: bool
Stereotype:
Description: Since reactive instances are performing second dispatching of the termination event the active object must follow the same policy to enable reactive parts to finalize their self-destruction.
This attribute is used for this aim.

Operation information for Class: OMThread

Operation name: ~OMThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMThread()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: OMThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMThread(char* name,long priority,long stackSize,long messageQueueSize,bool dynamicMessageQueue)
Return Type:
Description: Initialize the thread

Argument information for Operation OMThread

Name	Type	Direction
name	char*	In
priority	long	In
stackSize	long	In
messageQueueSize	long	In
dynamicMessageQueue	bool	In

Operation name: OMThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMThread(bool wrapThread)
Return Type:
Description: Initialize the thread

Argument information for Operation OMThread

Name	Type	Direction
wrapThread	bool	In

Operation name: queue

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: queue(IOxfEvent ev,IOxfEventGenerationParams params)
Return Type: bool
Description: Queue the event for later processing

Argument information for Operation queue

Name	Type	Direction
ev	IOxfEvent	In
params	IOxfEventGenerationParams	In

Operation name: endDispatching

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: endDispatching()
Return Type: void
Description: end the active instance event dispatching and destroy the instance.

Operation name: dispatch

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: dispatch(IOxfEvent ev)
Return Type: TakeEventStatus
Description: dispatch the specified event to its destination

Argument information for Operation dispatch

Name	Type	Direction
ev	IOxfEvent	In

Operation name: execute

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true

Visibility: protected
Signature: execute()
Return Type: OMReactive
Description: The thread event loop

Operation name: _cleanupThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _cleanupThread()
Return Type: void
Description: cleanup - called from the DTOR and from cleanupThread()

Operation name: _initializeOMThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _initializeOMThread(bool wrapThread,char* name,long stackSize,long messageQueueSize,bool dynamicMessageQueue)
Return Type: void
Description: initialization

Argument information for Operation _initializeOMThread

Name	Type	Direction
wrapThread	bool	In
name	char*	In
stackSize	long	In
messageQueueSize	long	In
dynamicMessageQueue	bool	In

Operation name: allowDeleteInThreadsCleanup

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: allowDeleteInThreadsCleanup()
Return Type: bool
Description: Check if the thread can be deleted

Operation name: cancelPendingEvent

Initializer:
Const: false

Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: cancelPendingEvent(IOxfEvent ev)
Return Type: void
Description: Cancel an invent event in the queue.
This operation requires an ability to iterate over the events waiting in the queue.

Argument information for Operation cancelPendingEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: cancelPendingEvents

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: cancelPendingEvents(IOxfReactive destination)
Return Type: void
Description: Cancel all events targeted for destination.
This operation requires an ability to iterate over the events waiting in the queue.

Argument information for Operation cancelPendingEvents

Name	Type	Direction
destination	IOxfReactive	In

Operation name: cleanupAllThreads

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: cleanupAllThreads()
Return Type: OMThread
Description: Destroy all the active threads

Operation name: cleanupThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected

Signature: cleanupThread()
Return Type: void
Description: cleanup - hook to allow cleanup of a thread without calling the DTOR
this method is needed to allow cleanup without destruction of the v-table

Operation name: destroyThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: destroyThread()
Return Type: void
Description: Destroy the active instance.

Operation name: doExecute

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: protected
Signature: doExecute(void * me)
Return Type: void
Description: task entry method - calls execute

Argument information for Operation doExecute

Name	Type	Direction
me	void *	In

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: get the OS thread handle

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public

Signature: getOsHandle(void * osHandle)
Return Type: void *
Description: Get the RTOS thread handle

Argument information for Operation getOsHandle

Name	Type	Direction
osHandle	void *	InOut

Operation name: getOSThreadEndClb

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getOSThreadEndClb(EndCallBack clb_p,void * arg1_p,bool onExecuteThread)
Return Type: void
Description: asking for a callback to end my os thread
onExecuteThread = true: I will kill my own thread
onExecuteThread = false: someone else will kill my thread

Argument information for Operation getOSThreadEndClb

Name	Type	Direction
clb_p	EndCallBack	Out
arg1_p	void *	Out
onExecuteThread	bool	In

Operation name: lock

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: lock()
Return Type: void
Description: Guard API

Operation name: resume

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: resume()
Return Type: void
Description: resume the thread

Operation name: setEndOSThreadInDtor

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setEndOSThreadInDtor(bool b)
Return Type: void
Description: set the os thread in DTOR flag

Argument information for Operation setEndOSThreadInDtor

Name	Type	Direction
b	bool	In

Operation name: setPriority

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setPriority(int pr)
Return Type: void
Description: set the thread priority

Argument information for Operation setPriority

Name	Type	Direction
pr	int	In

Operation name: startDispatching

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: startDispatching(bool /**/)
Return Type: void
Description: start the thread & the event loop -

IMPORTANT: OMThread ignore start parameter!!
the parameter should be checked only in default application threads (OMMainThread)

when creating an alternative default thread -
when doFork is set to false, the framework is expected to use the OS main thread.
when doFork is set to true is should create a new thread.

Argument information for Operation startDispatching

Name	Type	Direction
/**/	bool	In

Operation name: stopAllThreads

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: stopAllThreads(OMThread skipme)
Return Type: OMThread
Description: Stop the execution of all the framework threads to enable unload of framework related modules (used for COM applications support)

Argument information for Operation stopAllThreads

Name	Type	Direction
skipme	OMThread	In

Operation name: suspend

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: suspend()
Return Type: void
Description: suspend the thread

Operation name: unlock

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: unlock()
Return Type: void
Description: Unlock the thread guard

Operation name: getEventQueue

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false

Visibility: public
Signature: getEventQueue()
Return Type: OMEventQueue
Description: Get the thread event queue

Operation name: isCanceled

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: isCanceled(IOxfEvent ev)
Return Type: bool
Description: Check if the event is canceled

Argument information for Operation isCanceled

Name	Type	Direction
ev	IOxfEvent	In

Operation name: omGetEventQueue

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: omGetEventQueue()
Return Type: OMEventQueue
Description: Get the event queue of this thread

Operation name: unschedTm

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: unschedTm(ID id,OMReactive c)
Return Type: void
Description: Remove timeout

Argument information for Operation unschedTm

Name	Type	Direction
id	ID	In
c	OMReactive	In

Operation name: schedTm

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: schedTm(OxfTimeUnit deltaTime,short id,OMReactive instance,OMHandle state)
Return Type: void
Description: Set timeout

Argument information for Operation schedTm

Name	Type	Direction
deltaTime	OxfTimeUnit	In
id	short	In
instance	OMReactive	In
state	OMHandle	In

Operation name: queueEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: queueEvent(OMEvent ev,bool fromISR)
Return Type: bool
Description: Queue the event for later processing

Argument information for Operation queueEvent

Name	Type	Direction
ev	OMEvent	In
fromISR	bool	In

Operation name: cancelEvents

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: cancelEvents(OMReactive destination)
Return Type: void
Description: cancel all events targeted for destination

Argument information for Operation cancelEvents

Name	Type	Direction
destination	OMReactive	In

Operation name: cancelEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: cancelEvent(OMEvent ev)
Return Type: void
Description: cancel event

Argument information for Operation cancelEvent

Name	Type	Direction
ev	OMEvent	In

Operation name: start

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: start(int doFork)
Return Type: void
Description: backward compatibility

Argument information for Operation start

Name	Type	Direction
doFork	int	In

Operation name: shouldDispatch

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: shouldDispatch(IOxfEvent ev)
Return Type: bool
Description: Check if the event should be dispatched

Argument information for Operation shouldDispatch

Name	Type	Direction
ev	IOxfEvent	In

Operation name: isControlEvent

Initializer:

Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: isControlEvent(IOxfEvent ev)
Return Type: bool
Description: Check if the event is a framework control event

Argument information for Operation isControlEvent

Name	Type	Direction
ev	IOxfEvent	In

Operation name: setOsThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: setOsThread(OMOSThread* thread)
Return Type: void
Description: Set the OSAL thread

Argument information for Operation setOsThread

Name	Type	Direction
thread	OMOSThread*	In

Generalization information for Class OMThread

Generalization name: IOxfActive

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfActive	IOxfActive	OMThread

Relation information for Class OMThread

Relation name: dispatchingGuard

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: dispatchingGuard
LinkName:
RoleName: dispatchingGuard

Type: Composition
Description: The event dispatching guard - prevents mutual exclusion between the event dispatching and instance deletion.

Relation name: osThread

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: osThread
LinkName:
RoleName: osThread
Type: Composition
Description:

Relation name: eventQueue

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: eventQueue
LinkName:
RoleName: eventQueue
Type: Composition
Description: The thread event queue

Name	Inverse	Source	Target
dispatchingGuard		OMThread	OMProtected
osThread		OMThread	OMOSThread
eventQueue		OMThread	OMEventQueue

Class name: OMEvent

Description:The event base implementation class
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMEvent

Attribute Name: lId

Default Value: eventId
Static: false
Visibility: public
Type: ID
Stereotype:
Description: The event id.

Attribute Name: deleteAfterConsume

Default Value: true
Static: false
Visibility: public
Type: bool
Stereotype:

Description: This flag is used to indicate if the event dispatcher should delete the event after it was dispatched

Attribute Name: frameworkEvent

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: This flag mark an event as an internal framework event (that should be invisible in design-level debugging)

Attribute Name: port

Default Value: 0
Static: false
Visibility: public
Type: void *
Stereotype:
Description: The port that the event was sent to

Attribute Name: synchronous

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: Mark the event as a synchronous event (i.e. triggered operation).

Operation information for Class: OMEvent

Operation name: ~OMEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMEvent()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: OMEvent

Initializer: destination(dest)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMEvent(ID eventId,IOxfReactive dest)
Return Type:
Description: initialize the event

Argument information for Operation OMEvent

Name	Type	Direction
eventId	ID	In
dest	IOxfReactive	In

Operation name: destroy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: destroy()
Return Type: void
Description: destroy the event

Operation name: isTypeOf

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isTypeOf(ID eventId)
Return Type: bool
Description: check if the event is a sub-type of an event with the specified id

Argument information for Operation isTypeOf

Name	Type	Direction
eventId	ID	In

Operation name: isTimeout

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isTimeout()
Return Type: bool
Description: Check if the event is a timeout

Operation name: isRealEvent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false

Visibility: public
Signature: isRealEvent()
Return Type: bool
Description: Check that the event is an application event.

Operation name: Delete

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: Delete()
Return Type: void
Description: destroy the event if it should be destroyed

Operation name: operator delete

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator delete(void * /*object*/,void * /**/)
Return Type: void
Description: Dummy delete operator, added to avoid compilation warnings in some compilers (added to the compilation by definition of the NEED_DELETE_OPERATOR_FOR_STATIC_ALLOC flag)

Argument information for Operation operator delete

Name	Type	Direction
/*object*/	void *	In
/**/	void *	In

Operation name: setDueTime

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setDueTime(OxfTimeUnit /**/)
Return Type: void
Description: Empty implementation to IOxfTimeout API

Argument information for Operation setDueTime

Name	Type	Direction
/**/	OxfTimeUnit	In

Operation name: setDelayTime

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setDelayTime(OxfTimeUnit /**/)
Return Type: void
Description: Empty implementation to IOxfTimeout API

Argument information for Operation setDelayTime

Name	Type	Direction
/**/	OxfTimeUnit	In

Operation name: cancel

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cancel()
Return Type: void
Description: cancel the timeout

Operation name: isCanceled

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isCanceled()
Return Type: bool
Description: Empty implementation to ITimeout API

Operation name: getDueTime

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getDueTime()
Return Type: OxfTimeUnit
Description: Empty implementation to ITimeout API

Operation name: getDelayTime

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getDelayTime()
Return Type: OxfTimeUnit
Description: Empty implementation to IOxfTimeout API

Operation name: getIId

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getIId()
Return Type: ID
Description: compatibility API for getId()

Operation name: OMEvent

Initializer: lId(ev.lId), deleteAfterConsume(ev.deleteAfterConsume), frameworkEvent(ev.frameworkEvent), port(ev.port), synchronous(ev.synchronous), destination(ev.destination)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMEvent(OMEvent ev)
Return Type:
Description: copy constructor

Argument information for Operation OMEvent

Name	Type	Direction
ev	OMEvent	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(OMEvent ev)
Return Type: OMEvent
Description: assignment operator

Argument information for Operation operator=

Name	Type	Direction
ev	OMEvent	In

Operation name: isDeleteAfterConsume

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isDeleteAfterConsume()
Return Type: bool
Description:

Generalization information for Class OMEvent

Generalization name: IOxfTimeout

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfTimeout	IOxfTimeout	OMEvent

Relation information for Class OMEvent

Relation name: destination

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: destination
LinkName:
RoleName: destination
Type: Association
Description:

Name	Inverse	Source	Target
destination		OMEvent	IOxfReactive

Class name: OMTimeout

Description:IOxfTimeout implementation
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMTimeout

Attribute Name: canceled

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: When the event is canceled, it should be ignored by the event dispatcher.

Attribute Name: delayTime

Default Value: 0
Static: false
Visibility: public
Type: OxfTimeUnit
Stereotype:
Description: The relative delay until the timeout should be expired.

Attribute Name: dueTime

Default Value: 0
Static: false
Visibility: public
Type: OxfTimeUnit
Stereotype:
Description: The absolute time (i.e. system time) until the timeout will expire.
This time is calculated by:
dueTime = <timeout scheduling time> + delayTime.

Attribute Name: state

Default Value: 0
Static: false
Visibility: public
Type: OMHandle
Stereotype:
Description: The state that is the client of the timeout (for design level debugging)

Attribute Name: timeoutId

Default Value: 0
Static: false
Visibility: public
Type: short
Stereotype:
Description: the timeout id, exists to support extensive timeout management

Operation information for Class: OMTimeout

Operation name: ~OMTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: true
Visibility: public
Signature: ~OMTimeout()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: OMTimeout

Initializer: OMEvent(OMTimeoutEventId, pdest), canceled(false), delayTime(0) ,dueTime(0) ,state(0) ,timeoutId(0)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMTimeout(IOxfReactive pdest,OxfTimeUnit delay,OMHandle aomArg(theState))
Return Type:
Description: empty argument declaration, for arguments used in instrumented mode only.

Argument information for Operation OMTimeout

Name	Type	Direction
pdest	IOxfReactive	In
delay	OxfTimeUnit	In
aomArg(theState)	OMHandle	In

Operation name: OMTimeout

Initializer: OMEvent(OMTimeoutEventId, 0), canceled(false) ,delayTime(0) ,dueTime(0) ,state(0) ,timeoutId(0)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMTimeout()
Return Type:
Description: Initialize a timeout

Operation name: operator<

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator<(OMTimeout tn)
Return Type: bool
Description: Compare timeouts by due time

Argument information for Operation operator<

Name	Type	Direction
tn	OMTimeout	In

Operation name: operator>

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator>(OMTimeout tn)
Return Type: bool
Description: Compare timeouts by due time

Argument information for Operation operator>

Name	Type	Direction
tn	OMTimeout	In

Operation name: setRelativeDueTime

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setRelativeDueTime(OxfTimeUnit now)
Return Type: void
Description: Set the timeout due time based on the current time and the delay time

Argument information for Operation setRelativeDueTime

Name	Type	Direction
now	OxfTimeUnit	In

Operation name: cancel

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cancel()
Return Type: void
Description: cancel the timeout

Operation name: setDelayTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: public
Signature: setDelayTimeout()
Return Type: void
Description: set the timeout to be a delay timeout

Operation name: operator =

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator =(OMTimeout tm)
Return Type: OMTimeout
Description: assignment operator

Argument information for Operation operator =

Name	Type	Direction
tm	OMTimeout	In

Operation name: OMTimeout

Initializer: OMEvent(tm), canceled(tm.canceled), delayTime(tm.delayTime), dueTime(tm.dueTime), state(tm.state), timeoutId(tm.timeoutId)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMTimeout(OMTimeout tm)
Return Type:
Description: copy constructor

Argument information for Operation OMTimeout

Name	Type	Direction
tm	OMTimeout	In

Operation name: getDelay

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getDelay()
Return Type: OxfTimeUnit
Description: Backward compatibility: get the delay time

Generalization information for Class OMTimeout

Generalization name: OMEvent

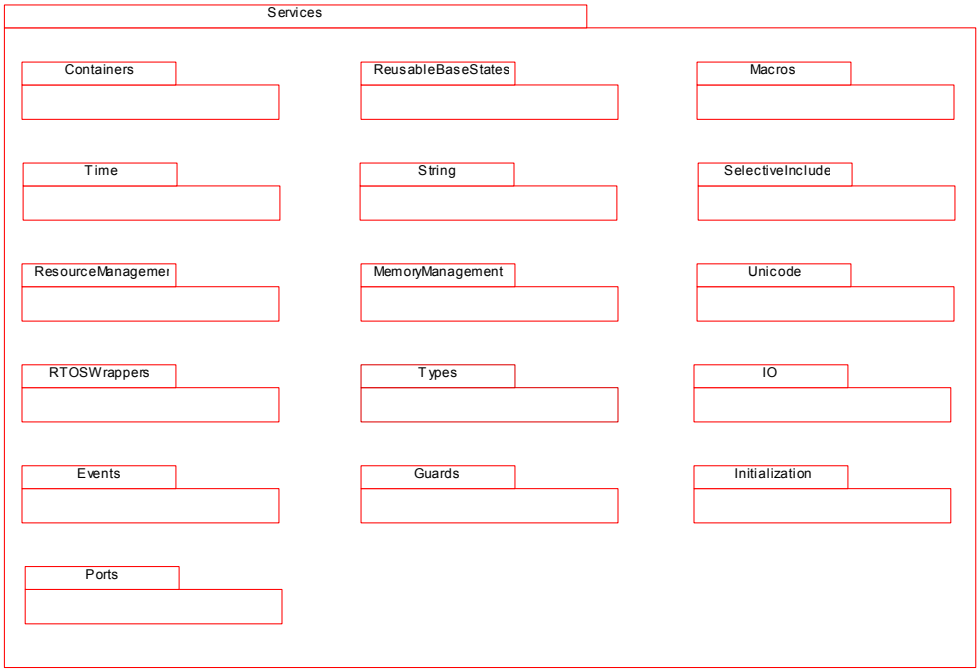
Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMEvent	OMEvent	OMTimeout

Package: Services

Object Model Diagram Information

Object Model Diagram name: Service packages overview
Description: Overview of the framework services



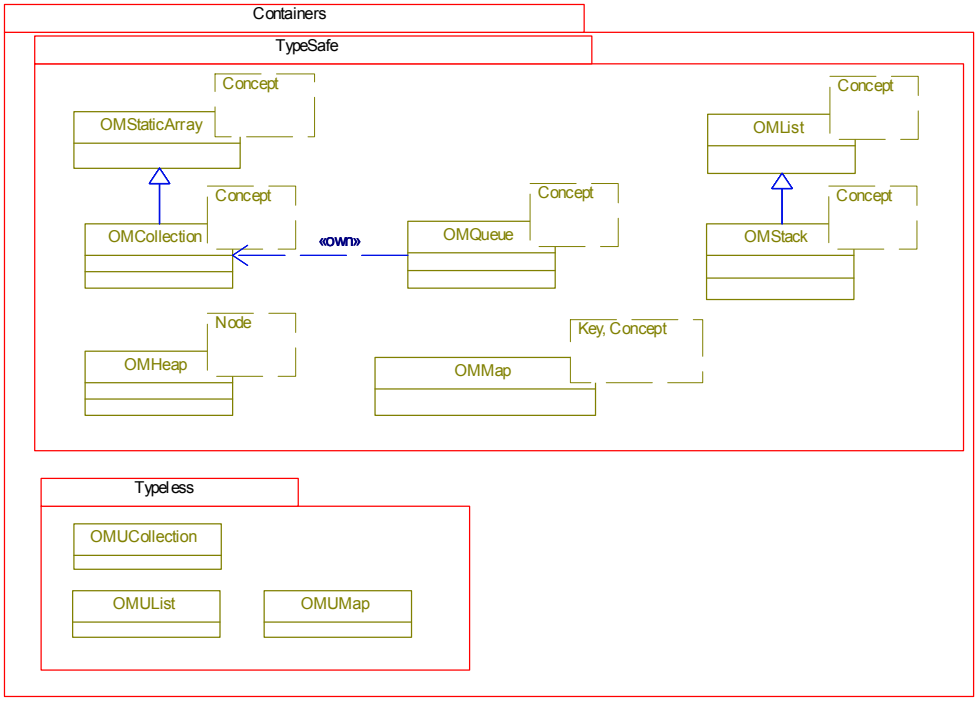
Package Information
Description: Services used by the framework and generated C++ applications.

Package: Containers

Object Model Diagram Information

Object Model Diagram name: Container Types

Description: The oxf container set (overview)



Package Information

Description: Container sets definitions

Package: STLContainersSupport

Class Information for Package: STLContainersSupport

Class name: OMValueCompare

Description:STL compare functor.
Used for qualified relations that are implemented with STL containers.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMValueCompare

Attribute Name: value_

Default Value: value
Static: false
Visibility: private
Type: Value
Stereotype: TemplateArgument
Description: The value to compare against (this is what we are looking for)

Operation information for Class: OMValueCompare

Operation name: OMValueCompare

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMValueCompare(Value value)
Return Type:
Description: constructor

Argument information for Operation OMValueCompare

Name	Type	Direction
value	Value	In

Operation name: operator()

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator()(const std::pair<Key,Value>& item)
Return Type: bool
Description: the operator used by std::find_if() to compare the map element with the one we are searching for

Argument information for Operation operator()

Name	Type	Direction
item	const std::pair<Key,Value>&	In

Type information for Class OMValueCompare

Type name: Value

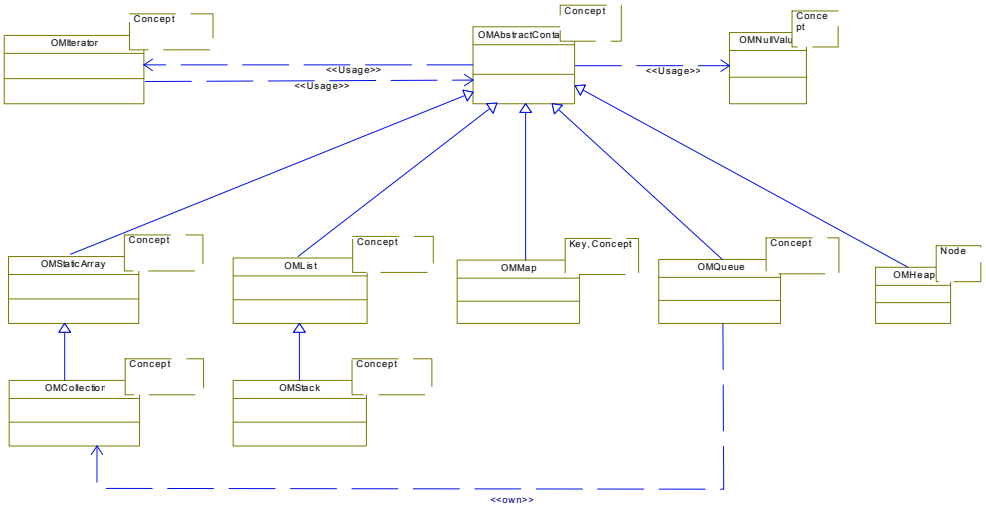
Description: Template argument modeling
Kind: Language

Package: TypeSafe

Object Model Diagram Information

Object Model Diagram name: type safe containers

Description: Overview of the type-safe containers



Class Information for Package: TypeSafe

Class name: OMMap

Description:A binary balanced tree map
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMMap

Attribute Name: count_

Default Value: 0
Static: false
Visibility: public
Type: unsigned long
Stereotype:
Description: The number of elements in the map

Attribute Name: root

Declaration: Item*
Default Value:
Static: false
Visibility: private
Type:
Stereotype:
Description: The tree root

Operation information for Class: OMMap

Operation name: ~OMMap

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: public
Signature: ~OMMap()
Return Type:
Description: Destructor

Operation name: add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: add(Key k,Concept p)
Return Type: void
Description: Add an element to the map using the provided key.

Argument information for Operation add

Name	Type	Direction
k	Key %s	In
p	Concept %s	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: copy(const OMMap<Key,Concept> & m)
Return Type: void
Description: Copy a map

Argument information for Operation copy

Name	Type	Direction
m	const OMMap<Key,Concept> & %s	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: copy(const Item* item)
Return Type: void
Description: Copy a sub tree

Argument information for Operation copy

Name	Type	Direction
item	const Item*	In

Operation name: find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: find(Concept p)
Return Type: int
Description: Find an element in the map

Argument information for Operation find

Name	Type	Direction
p	Concept %s	In

Operation name: getAt

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getAt(int i)
Return Type: Concept &
Description: Get an element from the map using an index

Argument information for Operation getAt

Name	Type	Direction
i	int	In

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getCurrent(void * pos)
Return Type: Concept &
Description: Get the element at the given position (called by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction

pos	void *	In

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getFirst(void*& pos)
Return Type: void
Description: Set the initial position for the iterator

Argument information for Operation getFirst

Name	Type	Direction
pos	void*& %s	Out

Operation name: getKey

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getKey(const Key& k)
Return Type: Concept &
Description: Get the element with the specified key

Argument information for Operation getKey

Name	Type	Direction
k	const Key& %s	In

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getNext(void*& pos)
Return Type: void
Description: Update the provided position to the next position in the container

Argument information for Operation getNext

Name	Type	Direction
pos	void*& %s	InOut

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isEmpty()
Return Type: int
Description: Check if the map is empty

Operation name: lookUp

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: lookUp(const Key k)
Return Type: Item*
Description: Find an item in the map based on a key

Argument information for Operation lookUp

Name	Type	Direction
k	const Key %s	In

Operation name: lookUp

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: lookUp(const Key k,Concept & c)
Return Type: int
Description: Find an element in the map based on a key
return 1 if found, 0 otherwise

Argument information for Operation lookUp

Name	Type	Direction
k	const Key %s	In
c	Concept & %s	Out

Operation name: OMMap

Initializer: root(NULL)

Const: false
Trigger: false
Abstract: false

Static: false
Virtual: false
Visibility: public
Signature: OMMap()
Return Type:
Description: Constructor

Operation name: operator[]

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator[](const Key & k)
Return Type: Concept &
Description: Get the element in the map using a key

Argument information for Operation operator[]

Name	Type	Direction
k	const Key & %s	In

Operation name: OMMap

Initializer: root(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMMap(const OMMap<Key, Concept>& m)
Return Type:
Description: copy constructor

Argument information for Operation OMMap

Name	Type	Direction
m	const OMMap<Key, Concept>&	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const OMMap<Key,Concept> & m)
Return Type: OMMap<Key,Concept> &
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
m	const OMMap<Key,Concept> & %s	In

Operation name: remove

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: remove(Concept p)
Return Type: void
Description: Remove an element from the map

Argument information for Operation remove

Name	Type	Direction
p	Concept %s	In

Operation name: remove

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: remove(Key k)
Return Type: void
Description: Remove a key from the map

Argument information for Operation remove

Name	Type	Direction
k	Key %s	In

Operation name: removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeAll()
Return Type: void
Description: Cleanup

Operation name: removeItem

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: removeItem(Item* item)
Return Type: void
Description: Remove an item from the map tree

Argument information for Operation removeItem

Name	Type	Direction
item	Item*	In

Generalization information for Class OMMap

Generalization name: OMAbstractContainer

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMAbstractContainer	OMAbstractContainer	OMMap

Class Information for Class: OMMap

Class name: Item

Description:A map item (node)
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: Item

Attribute Name: concept

Declaration: Concept
Default Value: theConcept
Static: false
Visibility: public
Type:
Stereotype:
Description: The item data

Attribute Name: key

Declaration: Key
Default Value: theKey
Static: false
Visibility: public

Type:
Stereotype:
Description: The item key

Attribute Name: larger

Declaration: Item*
Default Value: NULL
Static: false
Visibility: public
Type:
Stereotype:
Description: The right sub tree

Attribute Name: parent

Declaration: Item*
Default Value: NULL
Static: false
Visibility: public
Type:
Stereotype:
Description: The parent node in the tree

Attribute Name: rank

Default Value: 1
Static: false
Visibility: public
Type: int
Stereotype:
Description: The item rank in the balanced tree

Attribute Name: smaller

Declaration: Item*
Default Value: NULL
Static: false
Visibility: public
Type:
Stereotype:
Description: The left sub tree

Operation information for Class: Item

Operation name: _add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _add(Item* item)
Return Type: void
Description: Add an item to this sub tree

Argument information for Operation _add

Name	Type	Direction
item	Item*	In

Operation name: _addCheckBalance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _addCheckBalance()
Return Type: void
Description: Balance the sub tree

Operation name: _connect

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _connect(Item*& side,Item* item)
Return Type: void
Description: Connect the item to the sub tree

Argument information for Operation _connect

Name	Type	Direction
side	Item*&	Out
item	Item*	In

Operation name: _connectLarger

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _connectLarger(Item* item)
Return Type: void
Description: Connect the item to the right sub tree

Argument information for Operation _connectLarger

Name	Type	Direction
item	Item*	In

Operation name: _connectParent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _connectParent(Item* newN)
Return Type: void
Description: Connect the item to the parent sub tree

Argument information for Operation _connectParent

Name	Type	Direction
newN	Item*	In

Operation name: _connectSmaller

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _connectSmaller(Item* item)
Return Type: void
Description: Connect the item to the left sub tree

Argument information for Operation _connectSmaller

Name	Type	Direction
item	Item*	In

Operation name: _find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _find(const Concept & p)
Return Type: Item*
Description: Find the element in the sub tree

Argument information for Operation _find

Name	Type	Direction
p	const Concept & %s	In

Operation name: _lookUp

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _lookUp(const Key k)
Return Type: Item*
Description: Find a key in the sub tree - this is an O(log(N)) operation

Argument information for Operation _lookUp

Name	Type	Direction
k	const Key %s	In

Operation name: _removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _removeAll()
Return Type: void
Description: Cleanup the sub tree

Operation name: _removeCheckBalance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _removeCheckBalance()
Return Type: void
Description: Balance the sub tree after remove

Operation name: _removeYourSelf

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _removeYourSelf()
Return Type: void
Description: Remove this item from the map

Operation name: _switchNode

Initializer:
Const: false

Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _switchNode(Item* other)
Return Type: void
Description: Switch tree position with the specified item

Argument information for Operation _switchNode

Name	Type	Direction
other	Item*	In

Operation name: ~Item

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~Item()
Return Type:
Description: Cleanup

Operation name: Item

Initializer: parent(item.parent), smaller(item.smaller), larger(item.larger), concept(item.concept), key(item.key), rank(item.rank)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Item(const Item& item)
Return Type:
Description: Copy constructor

Argument information for Operation Item

Name	Type	Direction
item	const Item&	In

Operation name: Item

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Item(Key theKey,Concept theConcept)

Return Type:
Description: Initialize a map item with a key and data

Argument information for Operation Item

Name	Type	Direction
theKey	Key	In
theConcept	Concept	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const Item& item)
Return Type: Item&
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
item	const Item&	In

Operation name: getConcept

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getConcept()
Return Type: Concept&
Description: Get the item data

Class name: OMStack

Description:A stack (FILO)
Based on a linked-list
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMStack

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false

Static: false
Virtual: false
Visibility: private
Signature: copy(const OMStack<Concept>& s)
Return Type: void
Description: Copy a stack

Argument information for Operation copy

Name	Type	Direction
s	const OMStack<Concept>&	In

Operation name: OMStack

Initializer: OMList<Concept>()

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMStack(const OMStack<Concept> & s)
Return Type:
Description: Copy constructor

Argument information for Operation OMStack

Name	Type	Direction
s	const OMStack<Concept> & %s	In

Operation name: OMStack

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMStack()
Return Type:
Description: Initialize an empty stack

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const OMStack<Concept> & s)
Return Type: OMStack<Concept> &
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
s	const OMStack<Concept> & %s	In

Operation name: pop

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: pop()
Return Type: Concept
Description: Pop the top of the stack

Operation name: push

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: push(Concept p)
Return Type: void
Description: Push an element to the stack

Argument information for Operation push

Name	Type	Direction
p	Concept	In

Operation name: top

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: top()
Return Type: Concept
Description: Get the top of the stack without changing its state

Generalization information for Class OMStack

Generalization name: OMList

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMList	OMList	OMStack

Class name: OMStaticArray

Description:A fixed-size safe array
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMStaticArray

Attribute Name: count

Default Value: 0
Static: false
Visibility: public
Type: int
Stereotype:
Description: The number of elements currently placed in the array

Attribute Name: size

Default Value: 0
Static: false
Visibility: public
Type: int
Stereotype:
Description: The array total size

Attribute Name: theLink

Declaration: Concept*
Default Value: NULL
Static: false
Visibility: protected
Type:
Stereotype:
Description: The underlying C array

Operation information for Class: OMStaticArray

Operation name: ~OMStaticArray

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMStaticArray()
Return Type:
Description: Destructor

Operation name: add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: add(Concept p)
Return Type: void
Description: Add an element to the array

Argument information for Operation add

Name	Type	Direction
p	Concept	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: copy(const OMStaticArray<Concept> & a)
Return Type: void
Description: Copy an array

Argument information for Operation copy

Name	Type	Direction
a	const OMStaticArray<Concept> & %s	In

Operation name: find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: find(Concept p)
Return Type: bool
Description: Find if p in the collection

Argument information for Operation find

Name	Type	Direction
p	Concept	In

Operation name: getAt

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getAt(int i)
Return Type: Concept &
Description: Get the element in the specified index

Argument information for Operation getAt

Name	Type	Direction
i	int	In

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getCurrent(void * pos)
Return Type: Concept&
Description: Get the element at the given position (called by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getFirst(void * pos)
Return Type: void
Description: Set the initial position for the iterator

Argument information for Operation getFirst

Name	Type	Direction
pos	void *	Out

Operation name: getNext

Initializer:

Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getNext(void * pos)
Return Type: void
Description: Update the provided position to the next position in the container

Argument information for Operation getNext

Name	Type	Direction
pos	void *	InOut

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isEmpty()
Return Type: bool
Description: Check if the array is empty

Operation name: OMStaticArray

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMStaticArray(const OMStaticArray<Concept> & a)
Return Type:
Description: copy constructor

Argument information for Operation OMStaticArray

Name	Type	Direction
a	const OMStaticArray<Concept> & %s	In

Operation name: OMStaticArray

Initializer: count(0), size(theSize), theLink(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMStaticArray(int theSize)

Return Type:
Description: Initialize an array of a given size

Argument information for Operation OMStaticArray

Name	Type	Direction
theSize	int	In

Operation name: operator[]

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator[](int i)
Return Type: Concept &
Description: Get the element in the specified index

Argument information for Operation operator[]

Name	Type	Direction
i	int	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const OMStaticArray<Concept> & a)
Return Type: OMStaticArray<Concept> &
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
a	const OMStaticArray<Concept> & %s	In

Operation name: removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeAll()
Return Type: void
Description: Clear the array

Operation name: setAt

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setAt(int i,Concept c)
Return Type: void
Description: Set the element at the given index.
The index should be smaller then the number of elements currently located in the array

Argument information for Operation setAt

Name	Type	Direction
i	int	In
c	Concept	In

Generalization information for Class OMStaticArray

Generalization name: OMAbstractContainer

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMAbstractContainer	OMAbstractContainer	OMStaticArray

Class name: OMCollection

Description:A dynamic array
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMCollection

Operation name: add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: add(Concept p)
Return Type: void
Description: Add an element to the end of the array

Argument information for Operation add

Name	Type	Direction

p	Concept	In

Operation name: addAt

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: addAt(int index,Concept p)
Return Type: void
Description: add new element if in range, without increase of the container size.

Argument information for Operation addAt

Name	Type	Direction
index	int	In
p	Concept	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: copy(const OMCollection<Concept> & c)
Return Type: void
Description: Copy a collection

Argument information for Operation copy

Name	Type	Direction
c	const OMCollection<Concept> & %s	In

Operation name: OMCollection

Initializer: OMStaticArray<Concept>(0)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMCollection(const OMCollection<Concept> & c)
Return Type:
Description: copy constructor and assignment operator

Argument information for Operation OMCollection

Name	Type	Direction

c	const OMCollection<Concept> & %s	In

Operation name: OMCollection

Initializer: OMStaticArray<Concept>(theSize)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMCollection(int theSize)
Return Type:
Description: Constructor

Argument information for Operation OMCollection

Name	Type	Direction
theSize	int	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const OMCollection<Concept> & c)
Return Type: OMCollection<Concept> &
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
c	const OMCollection<Concept> & %s	In

Operation name: remove

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: remove(Concept p)
Return Type: void
Description: Remove p from the array

Argument information for Operation remove

Name	Type	Direction
p	Concept	In

Operation name: removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeAll()
Return Type: void
Description: Cleanup the array

Operation name: removeByIndex

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeByIndex(int i)
Return Type: void
Description: Remove the element at the specified index

Argument information for Operation removeByIndex

Name	Type	Direction
i	int	In

Operation name: reorganize

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: reorganize(int factor)
Return Type: void
Description: Reset the collection size

Argument information for Operation reorganize

Name	Type	Direction
factor	int	In

Operation name: reorgenize

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: public
Signature: reorgenize(int factor)
Return Type: void
Description: misspelled operation kept for backward compatibility

Argument information for Operation reorgenize

Name	Type	Direction
factor	int	In

Type information for Class OMCollection

Type name: Defaults

Description: Constant defaults used in the collection code
Kind: Enumeration

EnumerationLiteral information for Type Defaults

Name	Value
DEFAULT_START_SIZE	20
DEFAULT_FACTOR	2

Generalization information for Class OMCollection

Generalization name: OMStaticArray

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMStaticArray	OMStaticArray	OMCollection

Class name: OMQueue

Description:A FIFO queue
Implemented using a cyclic dynamic-array
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMQueue

Attribute Name: m_grow

Default Value:
Static: false
Visibility: protected
Type: bool
Stereotype:
Description: This flag indicates if the queue size should be dynamic (grow on demand) or static.

Attribute Name: m_head

Default Value:
Static: false
Visibility: protected
Type: int
Stereotype:
Description: The queue head (elements are extracted from the head)

Attribute Name: m_myQueue

Declaration: Collection
Default Value:
Static: false
Visibility: protected
Type:
Stereotype:
Description: The dynamic array used to implement the queue

Attribute Name: m_tail

Default Value:
Static: false
Visibility: protected
Type: int
Stereotype:
Description: The queue tail (elements are added to the tail)

Operation information for Class: OMQueue

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: copy(const OMQueue<Concept>& q)
Return Type: void
Description: Copy a queue

Argument information for Operation copy

Name	Type	Direction
q	const OMQueue<Concept>&	In

Operation name: get

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: get()
Return Type: Concept

Description: Get an element from the queue

Operation name: getCount

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getCount()
Return Type: int
Description: Get the number of elements in the queue

Operation name: getInverseQueue

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getInverseQueue(OMList<Concept> & l)
Return Type: void
Description: getQueue() returns the element which is the next to be returned by get() in the tail of the list

Argument information for Operation getInverseQueue

Name	Type	Direction
l	OMList<Concept> & %s	Out

Operation name: getQueue

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getQueue(OMList<Concept> & l)
Return Type: void
Description: getQueue() returns the element which is the next to be returned by get() in the head of the list

Argument information for Operation getQueue

Name	Type	Direction
l	OMList<Concept> & %s	Out

Operation name: getSize

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: public
Signature: getSize()
Return Type: int
Description: Get the size allocated for the queue

Operation name: increaseHead_

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: increaseHead_()
Return Type: void
Description: Update the queue head position

Operation name: increaseTail_

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: increaseTail_()
Return Type: void
Description: Advance the queue tail position and grow (if needed).

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isEmpty()
Return Type: bool
Description: Check if the queue is empty

Operation name: isFull

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isFull()
Return Type: bool
Description: Check if the queue is full.

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const OMQueue<Concept>& q)
Return Type: OMQueue<Concept>&
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
q	const OMQueue<Concept>&	In

Operation name: put

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: put(Concept c)
Return Type: bool
Description: Add an element to the queue

Argument information for Operation put

Name	Type	Direction
c	Concept %s	In

Operation name: OMQueue

Initializer: m_grow(shouldGrow), m_head(0), m_myQueue(initSize), m_tail(0)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMQueue(bool shouldGrow,int initSize)
Return Type:
Description: Initialize the queue with a given size and growth method (dynamic or static size)

Argument information for Operation OMQueue

Name	Type	Direction
shouldGrow	bool	In
initSize	int	In

Operation name: OMQueue

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMQueue(const OMQueue<Concept>& q)
Return Type:
Description: copy constructor

Argument information for Operation OMQueue

Name	Type	Direction
q	const OMQueue<Concept>&	In

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: getCurrent(void * pos)
Return Type: Concept&
Description: Get the element at the given position (called by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: getFirst(void*& pos)
Return Type: void
Description: Set the initial position for the iterator

Argument information for Operation getFirst

Name	Type	Direction
pos	void*& %s	Out

Operation name: getNext

Initializer:

Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: getNext(void*& pos)
Return Type: void
Description: Update the provided position to the next position in the container

Argument information for Operation getNext

Name	Type	Direction
pos	void*& %s	InOut

Generalization information for Class OMQueue

Generalization name: OMAbstractContainer

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMAbstractContainer	OMAbstractContainer	OMQueue

Class Information for Class: OMQueue

Class name: Collection

Description:The dynamic array type
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Class name: OMAbstractContainer

Description:A generic type-safe container, Used by OMIterator<> to iterate over the derived containers.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMAbstractContainer

Operation name: ~OMAbstractContainer

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMAbstractContainer()

Return Type:
Description: Virtual destructor to enable polymorphic deletion

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: private
Signature: getFirst(void*& pos)
Return Type: void
Description: Set the initial position for the iterator

Argument information for Operation getFirst

Name	Type	Direction
pos	void*&	Out

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: private
Signature: getNext(void*& pos)
Return Type: void
Description: Update the provided position to the next position in the container

Argument information for Operation getNext

Name	Type	Direction
pos	void*&	InOut

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: private
Signature: getCurrent(void * pos)
Return Type: Concept &
Description: Get the element at the given position (called by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Class name: OMHeap

Description://
template class OMHeap

The Heap invariants:
1. theHeap[0] is empty - this is so we have "easy arithmetic"
2. theHeap[1] -- theHeap[count] hold the actual elements
3. theHeap[i]<theHeap[2*i] && theHeap[i]<theHeap[2*i+1]
for all i>0, 2*i<=count, 2*i+1<=count.
Hence Min = theHeap[1]

OMHeap<Node> is a heap holding elements of type "Node*"

//
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMHeap

Attribute Name: theheap

Declaration: Node**
Default Value: NULL
Static: false
Visibility: private
Type:
Stereotype:
Description: An array of Node*'s - the data of the Heap

Attribute Name: count

Default Value: 0
Static: false
Visibility: public
Type: int
Stereotype:
Description: The number of items currently in the Heap

Attribute Name: heapSize

Default Value: 0
Static: false
Visibility: public
Type: int
Stereotype:
Description: The memory allocated for theHeap

Operation information for Class: OMHeap

Operation name: OMHeap

Initializer:
Const: false
Trigger: false
Abstract: false

Static: false
Virtual: false
Visibility: public
Signature: OMHeap(int size)
Return Type:
Description: Initialize a heap with a given size

Argument information for Operation OMHeap

Name	Type	Direction
size	int	In

Operation name: OMHeap

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMHeap(const OMHeap<Node> & h)
Return Type:
Description: copy constructor and assignment operator

Argument information for Operation OMHeap

Name	Type	Direction
h	const OMHeap<Node> & %s	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const OMHeap<Node> & h)
Return Type: OMHeap<Node> &
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
h	const OMHeap<Node> & %s	In

Operation name: ~OMHeap

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public

Signature: ~OMHeap()
Return Type:
Description: cleanup

Operation name: top

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: top()
Return Type: Node *
Description: Get the top of the heap

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isEmpty()
Return Type: int
Description: Check if the heap is empty

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getFirst(void * & pos)
Return Type: void
Description: Set the initial position for the iterator

Argument information for Operation getFirst

Name	Type	Direction
pos	void * & %s	Out

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getNext(void * & pos)

Return Type: void
Description: Update the provided position to the next position in the container

Argument information for Operation getNext

Name	Type	Direction
pos	void * & %s	InOut

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: private
Signature: getCurrent(void * pos)
Return Type: Node&
Description: Get the element at the given position (called by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: copy(const OMHeap<Node> & h)
Return Type: void
Description: Copy a heap

Argument information for Operation copy

Name	Type	Direction
h	const OMHeap<Node> & %s	In

Operation name: find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: find(Node* clone)
Return Type: int
Description: Return the position of the first e such that
(*e) == (*clone) return 0 if not found

Argument information for Operation find

Name	Type	Direction
clone	Node*	In

Operation name: takeUp

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: takeUp(Node* e,int emptyPos)
Return Type: void
Description: e is a node which needs to be placed in the heap.
Its position can be either 'emptyPos' which is currently empty or some position higher than emptyPos (if elements currently there are bigger than e)

Argument information for Operation takeUp

Name	Type	Direction
e	Node*	In
emptyPos	int	In

Operation name: takeDown

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: takeDown(Node* e,int emptyPos)
Return Type: void
Description: e is a node which needs to be placed in the heap.
Its position can be either 'emptyPos' which is currently empty or some position lower than emptyPos (if elements currently there are smaller than e(

Argument information for Operation takeDown

Name	Type	Direction
e	Node*	In
emptyPos	int	In

Operation name: add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: add(Node* e)

Return Type: void
Description: Add e to heap

Argument information for Operation add

Name	Type	Direction
e	Node*	In

Operation name: remove

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: remove(Node* clone)
Return Type: Node *
Description: Remove the first e such that
(*e) == (*clone)
return e if found NULL otherwise

Argument information for Operation remove

Name	Type	Direction
clone	Node*	In

Operation name: trim

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: trim()
Return Type: void
Description: Remove top

Operation name: removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeAll()
Return Type: void
Description: Cleanup the heap

Generalization information for Class OMHeap

Generalization name: OMAbstractContainer

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMAbstractContainer	OMAbstractContainer	OMHeap

Class name: OMIterator

Description:Iterator on containers derived from OMAbstractContainer
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMIterator

Attribute Name: thePos

Default Value:
Static: false
Visibility: private
Type: void *
Stereotype:
Description: The current position of the iterator (in the collection)

Operation information for Class: OMIterator

Operation name: OMIterator

Initializer: theLink(0)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMIterator()
Return Type:
Description: Initialize an empty iterator

Operation name: OMIterator

Initializer: theLink(&l)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMIterator(const Container& l)
Return Type:
Description: Initialize an iterator associated with the provided container

Argument information for Operation OMIterator

Name	Type	Direction
l	const Container&	In

Operation name: OMIterator

Initializer: theLink(l)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMIterator(const Container* l)
Return Type:
Description: Initialize an iterator associated with the provided container

Argument information for Operation OMIterator

Name	Type	Direction
l	const Container*	In

Operation name: OMIterator

Initializer: thePos(iter.thePos), theLink(iter.theLink)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMIterator(const OMIterator<Concept> & iter)
Return Type:
Description: Copy constructor

Argument information for Operation OMIterator

Name	Type	Direction
iter	const OMIterator<Concept> & %s	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const OMIterator<Concept> & iter)
Return Type: OMIterator<Concept> &
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
iter	const OMIterator<Concept> & %s	In

Operation name: increment

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: increment()
Return Type: OMIterator<Concept> &
Description: Move to the next item in the collection

Operation name: value

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: value()
Return Type: Concept &
Description: Get the current item

Operation name: operator*

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator*()
Return Type: Concept &
Description: Get the current item

Operation name: reset

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: reset()
Return Type: void
Description: Reset the iterator (to the first element in the container)

Operation name: reset

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: reset(const Container& newLink)
Return Type: void
Description: Reset the iterator to the specified container

Argument information for Operation reset

Name	Type	Direction
newLink	const Container&	In

Operation name: operator++

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator++()
Return Type: OMIterator<Concept> &
Description: Advance to the next item in the collection

Operation name: operator++

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator++(int /**/)
Return Type: OMIterator<Concept>
Description: Advance to the next item in the collection (postfix operator)

Argument information for Operation operator++

Name	Type	Direction
/**/	int	In

Operation name: _advance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private

Signature: _advance()
Return Type: void
Description: advance to the next item in the context collection

Operation name: ~OMIterator

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMIterator()
Return Type:
Description: Cleanup

Relation information for Class OMIterator

Relation name: theLink

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theLink
LinkName:
RoleName: theLink
Type: Association
Description:

Name	Inverse	Source	Target
theLink		OMIterator	Container

Class Information for Class: OMIterator

Class name: Container

Description:Abstract container instantiation used by the iterator
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Class name: OMList

Description:Linked list container class
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMList

Attribute Name: count_

Default Value: 0
Static: false

Visibility: public
Type: unsigned long
Stereotype:
Description: the number of elements in the list

Attribute Name: first

Declaration: Item*
Default Value: NULL
Static: false
Visibility: private
Type:
Stereotype:
Description: the list head

Attribute Name: last

Declaration: Item*
Default Value: NULL
Static: false
Visibility: private
Type:
Stereotype:
Description: the tail head

Operation information for Class: OMList

Operation name: OMList

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMList()
Return Type:
Description: Constructor - create an empty list

Operation name: OMList

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMList(const OMList<Concept> & l)
Return Type:
Description: copy constructor

Argument information for Operation OMList

Name	Type	Direction
l	const OMList<Concept> & %s	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const OMList<Concept> & l)
Return Type: OMList<Concept> &
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
l	const OMList<Concept> & %s	In

Operation name: ~OMList

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMList()
Return Type:
Description: Destructor - empty the list

Operation name: removeFirst

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeFirst()
Return Type: void
Description: Remove first item from list

Operation name: removeLast

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeLast()
Return Type: void
Description: Remove last item from list - inefficient as we keep no
backward pointers

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isEmpty()
Return Type: bool
Description: Check if the list is empty

Operation name: getFirstConcept

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getFirstConcept()
Return Type: Concept &
Description: Get the element in the head of the list

Operation name: getLastConcept

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getLastConcept()
Return Type: Concept &
Description: Get the element in the tail of the list

Operation name: operator[]

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator[](int i)
Return Type: Concept &
Description: Get the element at the provided index

Argument information for Operation operator[]

Name	Type	Direction
i	int	In

Operation name: _removeFirst

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: _removeFirst()
Return Type: void
Description: unsafe Remove first item from list

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: getFirst(void*& pos)
Return Type: void
Description: Set the initial position for the iterator

Argument information for Operation getFirst

Name	Type	Direction
pos	void*& %s	Out

Operation name: getLast

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: getLast(void*& pos)
Return Type: void
Description: Bet the position of the list tail

Argument information for Operation getLast

Name	Type	Direction
pos	void*& %s	Out

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected

Signature: getNext(void*& pos)
Return Type: void
Description: Update the provided position to the next position in the container

Argument information for Operation getNext

Name	Type	Direction
pos	void*& %s	InOut

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: getCurrent(void * pos)
Return Type: Concept &
Description: Get the element at the given position (called by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: copy(const OMList<Concept> & l)
Return Type: void
Description: Copy the specified list

Argument information for Operation copy

Name	Type	Direction
l	const OMList<Concept> & %s	In

Operation name: removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeAll()
Return Type: void
Description: Remove all items from list

Operation name: find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: find(Concept c)
Return Type: bool
Description: Find an object in the list

Argument information for Operation find

Name	Type	Direction
c	Concept %s	In

Operation name: getAt

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getAt(int i)
Return Type: Concept &
Description: Get the element in the given index

Argument information for Operation getAt

Name	Type	Direction
i	int	In

Operation name: add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: add(Concept c)
Return Type: void
Description: Add an object to the list (at its end)

Argument information for Operation add

Name	Type	Direction
c	Concept %s	In

Operation name: addFirst

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: addFirst(Concept c)
Return Type: void
Description: Add an object to the list (at its beginning)

Argument information for Operation addFirst

Name	Type	Direction
c	Concept %s	In

Operation name: addAt

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: addAt(int i,Concept c)
Return Type: void
Description: if getCount>=i - Add c after i'th element else Add c at end

Argument information for Operation addAt

Name	Type	Direction
i	int	In
c	Concept %s	In

Operation name: removeItem

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeItem(Item* item)
Return Type: void
Description: Remove a specific Item from list

Argument information for Operation removeItem

Name	Type	Direction
item	Item*	In

Operation name: remove

Initializer:
Const: false
Trigger: false

Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: remove(Concept c)
Return Type: void
Description: Remove the first occurrence of a specific Object (Concept)
from list

Argument information for Operation remove

Name	Type	Direction
c	Concept %s	In

Generalization information for Class OMList

Generalization name: OMAbstractContainer

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMAbstractContainer	OMAbstractContainer	OMList

Class Information for Class: OMList

Class name: Item

Description:List node
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: Item

Attribute Name: next

Declaration: Item*
Default Value:
Static: false
Visibility: public
Type:
Stereotype:
Description: the next item in the list

Attribute Name: concept

Declaration: Concept
Default Value:
Static: false
Visibility: public
Type:
Stereotype:
Description: the data of the node

Operation information for Class: Item

Operation name: Item

Initializer: concept(theConcept), next(NULL)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Item(const Concept& theConcept)
Return Type:
Description: Initialize an item with a given data

Argument information for Operation Item

Name	Type	Direction
theConcept	const Concept&	In

Operation name: Item

Initializer: concept(other.concept), next(other.next)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Item(const Item& other)
Return Type:
Description: Copy constructor

Argument information for Operation Item

Name	Type	Direction
other	const Item&	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const Item& other)
Return Type: Item&
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
other	const Item&	In

Operation name: connectTo

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: connectTo(Item* item)
Return Type: void
Description: Connect to the specified item

Argument information for Operation connectTo

Name	Type	Direction
item	Item*	In

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getNext()
Return Type: Item*
Description: Get the next item in the linked list

Operation name: ~Item

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~Item()
Return Type:
Description: Cleanup

Class name: OMNullValue

Description:Empty value class - used by containers to return non-existing element
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMNullValue

Operation name: get

Initializer:

Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: get()
Return Type: Concept &
Description: Get the empty value

Type information for Package TypeSafe

Type name: ContainersMemoryAlignment

Description: When allocating memory, the memory blocks must be aligned with the compiler settings. This macro is used in the memory pools, to guarantee that the pool allocates sufficient memory in order to avoid memory alignment issues.
Kind: Language
Declaration: #ifndef OM_LONG_MEMORY_ALIGNMENT
typedef char omMemoryAlignedType;
#define OM_ALIGNED_SIZEOF(ELEMENT) (sizeof(ELEMENT) +
(OMRAW_MEMORY_ALIGNMENT-1))
#else
typedef int omMemoryAlignedType;
#define OM_ALIGNED_SIZEOF(ELEMENT) ((sizeof(ELEMENT) +
(OMRAW_MEMORY_ALIGNMENT-1) + 3) / 4)
#endif // OM_LONG_MEMORY_ALIGNMENT

Attribute information for Package TypeSafe

Attribute name: OMContainersNullBlock

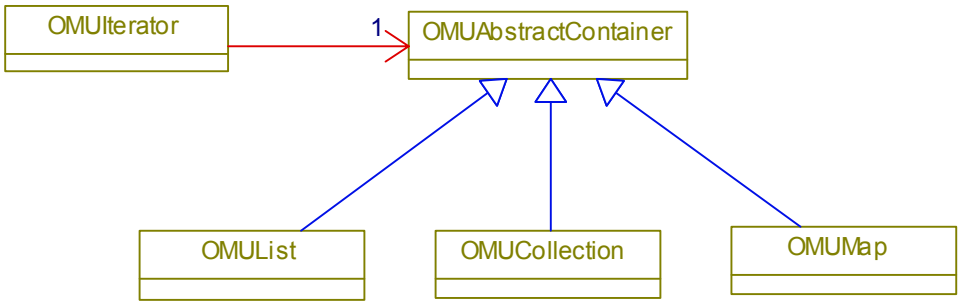
Type: int %s[]
Stereotype:
Declaration: int %s[]
Default Value: {0,0,0,0}
Description: NULL array used for null element (OMNullValue)

Package: Typeless

Object Model Diagram Information

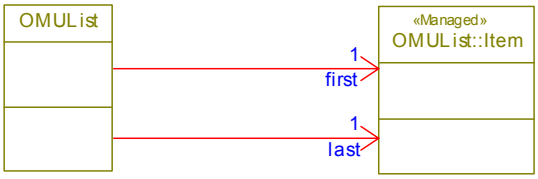
Object Model Diagram name: Typeless containers

Description: The type less containers



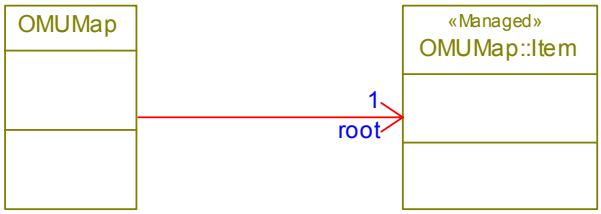
Object Model Diagram name: List

Description: OMUList structure



Object Model Diagram name: Map

Description: OMUMap overview



Class Information for Package: Typeless

Class name: OMUList

Description:type less list
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMUList

Operation name: _removeFirst

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: _removeFirst()
Return Type: void
Description: Remove first item from list

Operation name: ~OMUList

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMUList()
Return Type:
Description: Destructor - empty the list

Operation name: add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: add(void * p)
Return Type: void
Description: Add an object to the list (at its end)

Argument information for Operation add

Name	Type	Direction
p	void *	In

Operation name: addAt

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: addAt(int i,void * p)
Return Type: void
Description: if getCount() >= i - Add p after i'th element else Add p at end

Argument information for Operation addAt

Name	Type	Direction
i	int	In

p	void *	In

Operation name: addFirst

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: addFirst(void * p)
Return Type: void
Description: Add an object to the list (at its beginning)

Argument information for Operation addFirst

Name	Type	Direction
p	void *	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: copy(OMUList l)
Return Type: void
Description: Copy a list

Argument information for Operation copy

Name	Type	Direction
l	OMUList	In

Operation name: find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: find(const void * p)
Return Type: bool
Description: Find an object in the list, return 1 if found or 0 otherwise

Argument information for Operation find

Name	Type	Direction
p	const void * %s	In

Operation name: getAt

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getAt(int i)
Return Type: void *
Description: return the element in a given index

Argument information for Operation getAt

Name	Type	Direction
i	int	In

Operation name: getCount

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getCount()
Return Type: int
Description: return the number of elements in the list

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isEmpty()
Return Type: bool
Description: return true if the list is empty and false otherwise

Operation name: OMUList

Initializer: first(NULL), last(NULL)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUList(OMUList l)
Return Type:
Description: copy constructor and assignment operator

Argument information for Operation OMUList

Name	Type	Direction
l	OMUList	In

Operation name: OMUList

Initializer: first(NULL), last(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUList()
Return Type:
Description: Constructor - create an empty list

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(OMUList l)
Return Type: OMUList
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
l	OMUList	In

Operation name: remove

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: remove(void * p)
Return Type: void
Description: Remove the first occurrence of a specific Object from list

Argument information for Operation remove

Name	Type	Direction
p	void *	In

Operation name: removeAll

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeAll()
Return Type: void
Description: Remove all items from list

Operation name: removeFirst

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeFirst()
Return Type: void
Description: Remove first item from list

Operation name: removeItem

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeItem(Item item)
Return Type: void
Description: Remove a specific Item from list

Argument information for Operation removeItem

Name	Type	Direction
item	Item	In

Operation name: removeLast

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeLast()
Return Type: void
Description: Remove last item from list - inefficient as we keep no backward pointers

Operation name: operator[]

Initializer:

Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator[](int i)
Return Type: void *
Description: return the element at the given index

Argument information for Operation operator[]

Name	Type	Direction
i	int	In

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: getCurrent(void * pos)
Return Type: void *
Description: Get the item in the provided position (supplied by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: getFirst(void * pos)
Return Type: void
Description: iteration interface - get the initial position

Argument information for Operation getFirst

Name	Type	Direction
pos	void *	InOut

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: false

Static: false
Virtual: false
Visibility: protected
Signature: getNext(void * pos)
Return Type: void
Description: Get the next position for the iterator

Argument information for Operation getNext

Name	Type	Direction
pos	void *	InOut

Generalization information for Class OMUList

Generalization name: OMUAbstractContainer

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMUAbstractContainer	OMUAbstractContainer	OMUList

Relation information for Class OMUList

Relation name: first

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: first
LinkName:
RoleName: first
Type: Association
Description:

Relation name: last

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: last
LinkName:
RoleName: last
Type: Association
Description:

Name	Inverse	Source	Target
first		OMUList	Item
last		OMUList	Item

Class Information for Class: OMUList

Class name: Item

Description:A list node
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: Item

Attribute Name: element

Default Value:
Static: false
Visibility: public
Type: void *
Stereotype:
Description: The element (data)

Attribute Name: next

Declaration: Item*
Default Value:
Static: false
Visibility: public
Type:
Stereotype:
Description: The next item

Operation information for Class: Item

Operation name: connectTo

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: connectTo(Item* item)
Return Type: void
Description: connect to another item

Argument information for Operation connectTo

Name	Type	Direction
item	Item*	In

Operation name: Item

Initializer: element(item.element), next(item.next)
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: public
Signature: Item(const Item& item)
Return Type:
Description: copy constructor

Argument information for Operation Item

Name	Type	Direction
item	const Item&	In

Operation name: Item

Initializer: element(theElement), next(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Item(void * theElement)
Return Type:
Description: Constructor

Argument information for Operation Item

Name	Type	Direction
theElement	void *	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const Item& item)
Return Type: Item&
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
item	const Item&	In

Operation name: ~Item

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~Item()

Return Type:
Description: Cleanup

Class name: OMUMap

Description:A type less map - implemented as a balanced tree (log(N) search)
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMUMap

Operation name: ~OMUMap

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMUMap()
Return Type:
Description: Destructor

Operation name: add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: add(void * theKey,void * p)
Return Type: void
Description: add a new element to the given key

Argument information for Operation add

Name	Type	Direction
theKey	void *	In
p	void *	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: copy(OMUMap m)
Return Type: void
Description: copy a map

Argument information for Operation copy

Name	Type	Direction
m	OMUMap	In

Operation name: copy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: copy(Item item)
Return Type: void
Description: copy an item into the map

Argument information for Operation copy

Name	Type	Direction
item	Item	In

Operation name: find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: find(void * p)
Return Type: bool
Description: find an element in the map

Argument information for Operation find

Name	Type	Direction
p	void *	In

Operation name: getAt

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getAt(void * theKey)
Return Type: void *
Description: get the element for a given key

Argument information for Operation getAt

Name	Type	Direction

theKey	void *	In

Operation name: getCount

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getCount()
Return Type: int
Description: get the number of elements in the map

Operation name: getKey

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getKey(void * theKey)
Return Type: void *
Description: get the element for a given key

Argument information for Operation getKey

Name	Type	Direction
theKey	void *	In

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isEmpty()
Return Type: bool
Description: check for empty map

Operation name: lookUp

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: lookUp(void * theKey)
Return Type: Item

Description: return a map item for a given key

Argument information for Operation lookUp

Name	Type	Direction
theKey	void *	In

Operation name: lookUp

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: lookUp(void * theKey,void * element)
Return Type: bool
Description: find an element by its key
Place in "element" the element referenced by "theKey".
return true if found false otherwise

Argument information for Operation lookUp

Name	Type	Direction
theKey	void *	In
element	void *	Out

Operation name: OMUMap

Initializer: root(NULL)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUMap(OMUMap m)
Return Type:
Description: copy constructor and assignment operator

Argument information for Operation OMUMap

Name	Type	Direction
m	OMUMap	In

Operation name: OMUMap

Initializer: root(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUMap()

Return Type:
Description: Constructor

Operation name: operator[]

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator[](void * theKey)
Return Type: void *
Description: get the element for a given key

Argument information for Operation operator[]

Name	Type	Direction
theKey	void *	In

Operation name: remove

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: remove(Item item)
Return Type: void
Description: remove a map item

Argument information for Operation remove

Name	Type	Direction
item	Item	In

Operation name: remove

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: remove(void * p)
Return Type: void
Description: remove an element from the map

Argument information for Operation remove

Name	Type	Direction
p	void *	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(OMUMap m)
Return Type: OMUMap
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
m	OMUMap	In

Operation name: removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeAll()
Return Type: void
Description: remove all elements in the map

Operation name: removeKey

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeKey(void * theKey)
Return Type: void
Description: remove an element from the map by its key

Argument information for Operation removeKey

Name	Type	Direction
theKey	void *	In

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected

Signature: getCurrent(void * pos)
Return Type: void *
Description: Get the item in the provided position (supplied by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: getFirst(void * pos)
Return Type: void
Description: iteration interface - get the initial position

Argument information for Operation getFirst

Name	Type	Direction
pos	void *	InOut

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: getNext(void * pos)
Return Type: void
Description: Get the next position for the iterator

Argument information for Operation getNext

Name	Type	Direction
pos	void *	InOut

Generalization information for Class OMUMap

Generalization name: OMUAbstractContainer

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMUAbstractContainer	OMUAbstractContainer	OMUMap

Relation information for Class OMUMap

Relation name: root

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: root
LinkName:
RoleName: root
Type: Association
Description:

Name	Inverse	Source	Target
root		OMUMap	Item

Class Information for Class: OMUMap

Class name: Item

Description:A map item
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: Item

Attribute Name: element

Default Value: theElement
Static: false
Visibility: public
Type: void *
Stereotype:
Description: The item data

Attribute Name: key

Default Value: theKey
Static: false
Visibility: public
Type: void *
Stereotype:
Description: The key

Attribute Name: rank

Default Value: 1
Static: false
Visibility: public
Type: int
Stereotype:
Description: This node rank in the balanced tree

Attribute Name: larger

Declaration: Item*

Default Value: NULL
Static: false
Visibility: public
Type:
Stereotype:
Description: The right sub tree

Attribute Name: parent

Declaration: Item*
Default Value: NULL
Static: false
Visibility: public
Type:
Stereotype:
Description: The parent node

Attribute Name: smaller

Declaration: Item*
Default Value: NULL
Static: false
Visibility: public
Type:
Stereotype:
Description: The left sub tree

Operation information for Class: Item

Operation name: _add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _add(Item* item)
Return Type: void
Description: add a new map item to sub tree, and balance the sub tree

Argument information for Operation _add

Name	Type	Direction
item	Item*	In

Operation name: _addCheckBalance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _addCheckBalance()
Return Type: void

Description: balance the tree

Operation name: _connect

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _connect(Item*& side,Item* item)
Return Type: void
Description: connect sub trees

Argument information for Operation _connect

Name	Type	Direction
side	Item*&	Out
item	Item*	In

Operation name: _connectLarger

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _connectLarger(Item* item)
Return Type: void
Description: connect to the right sub tree

Argument information for Operation _connectLarger

Name	Type	Direction
item	Item*	In

Operation name: _connectParent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _connectParent(Item* newN)
Return Type: void
Description: connect to parent sub tree

Argument information for Operation _connectParent

Name	Type	Direction
newN	Item*	In

Operation name: _connectSmaller

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _connectSmaller(Item* item)
Return Type: void
Description: connect to the left sub tree

Argument information for Operation _connectSmaller

Name	Type	Direction
item	Item*	In

Operation name: _find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _find(void * p)
Return Type: const Item*
Description: find an element in the map item sub tree

Argument information for Operation _find

Name	Type	Direction
p	void *	In

Operation name: _getCount

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _getCount()
Return Type: int
Description: get the number of elements in the map item sub tree

Operation name: _lookUp

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private

Signature: _lookUp(void * theKey)
Return Type: Item*
Description: find a map item in this map item sub tree

Argument information for Operation _lookUp

Name	Type	Direction
theKey	void *	In

Operation name: _removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _removeAll()
Return Type: void
Description: remove all the elements in the map item sub tree

Operation name: _removeCheckBalance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _removeCheckBalance()
Return Type: void
Description: balance tree

Operation name: _removeYourSelf

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _removeYourSelf()
Return Type: void
Description: remove this map item

Operation name: _switchNode

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _switchNode(Item* other)

Return Type: void
Description: replace positions with the other map item

Argument information for Operation _switchNode

Name	Type	Direction
other	Item*	In

Operation name: ~Item

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~Item()
Return Type:
Description: Cleanup

Operation name: Item

Initializer: element(item.element), key(item.key), rank(item.rank), parent(item.parent), smaller(item.smaller), larger(item.larger)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Item(const Item& item)
Return Type:
Description: Copy constructor

Argument information for Operation Item

Name	Type	Direction
item	const Item&	In

Operation name: Item

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Item(void * theKey,void * theElement)
Return Type:
Description: Constructor

Argument information for Operation Item

Name	Type	Direction
theKey	void *	In

theElement	void *	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(const Item& item)
Return Type: Item&
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
item	const Item&	In

Class name: OMUCollection

Description:A type less dynamic array
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMUCollection

Attribute Name: count

Default Value: 0
Static: false
Visibility: public
Type: int
Stereotype:
Description: The number of elements stored in the array

Attribute Name: size

Default Value: 0
Static: false
Visibility: public
Type: int
Stereotype:
Description: The array allocated size

Attribute Name: theLink

Default Value: NULL
Static: false
Visibility: private
Type: void *
Stereotype:
Description: The raw array

Operation information for Class: OMUCollection

Operation name: ~OMUCollection

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMUCollection()
Return Type:
Description: Destructor

Operation name: add

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: add(void * p)
Return Type: void
Description: add an element to the collection, grow if need to

Argument information for Operation add

Name	Type	Direction
p	void *	In

Operation name: addAt

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: addAt(int index,void * p)
Return Type: int
Description: Add new element if in range, without increasing the collection size.
Return 0 if fail, or 1 on success.

Argument information for Operation addAt

Name	Type	Direction
index	int	In
p	void *	In

Operation name: copy

Initializer:
Const: false

Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: copy(OMUCollection c)
Return Type: void
Description: Copy a collection

Argument information for Operation copy

Name	Type	Direction
c	OMUCollection	In

Operation name: find

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: find(void * p)
Return Type: int
Description: If p in the collection return 1 else return 0

Argument information for Operation find

Name	Type	Direction
p	void *	In

Operation name: getAt

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getAt(int i)
Return Type: void *
Description: return the element at index i, if i is out of range, return 0;

Argument information for Operation getAt

Name	Type	Direction
i	int	In

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: protected
Signature: getFirst(void * pos)
Return Type: void
Description: iteration interface

Argument information for Operation getFirst

Name	Type	Direction
pos	void *	InOut

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: getCurrent(void * pos)
Return Type: void *
Description: Get the item in the provided position (supplied by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: getNext(void * pos)
Return Type: void
Description: Get the next position for the iterator

Argument information for Operation getNext

Name	Type	Direction
pos	void *	InOut

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public

Signature: isEmpty()
Return Type: bool
Description: return true if the collection is empty, else return false

Operation name: OMUCollection

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUCollection(OMUCollection c)
Return Type:
Description: copy constructor and assignment operator

Argument information for Operation OMUCollection

Name	Type	Direction
c	OMUCollection	In

Operation name: OMUCollection

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUCollection(int theSize)
Return Type:
Description: Constructor

Argument information for Operation OMUCollection

Name	Type	Direction
theSize	int	In

Operation name: operator[]

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator[](int i)
Return Type: void *
Description: operator [] - return the element in index i, or NULL if out of range

Argument information for Operation operator[]

Name	Type	Direction
i	int	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(OMUCollection c)
Return Type: OMUCollection
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
c	OMUCollection	In

Operation name: remove

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: remove(void * p)
Return Type: void
Description: remove p from the collection

Argument information for Operation remove

Name	Type	Direction
p	void *	In

Operation name: removeAll

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: removeAll()
Return Type: void
Description: clean up, and reset

Operation name: removeByIndex

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public

Signature: removeByIndex(int i)
Return Type: void
Description: remove the element in index i

Argument information for Operation removeByIndex

Name	Type	Direction
i	int	In

Operation name: reorganize

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: reorganize(int factor)
Return Type: void
Description: reorganize the collection, and grow if need to

Argument information for Operation reorganize

Name	Type	Direction
factor	int	In

Operation name: reorgenize

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: reorgenize(int factor)
Return Type: void
Description: misspelled operation kept for backward compatibility

Argument information for Operation reorgenize

Name	Type	Direction
factor	int	In

Operation name: setAt

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setAt(int i,void * p)
Return Type: int
Description: set an p in i

return 0 if fail, or 1 on success

Argument information for Operation setAt

Name	Type	Direction
i	int	In
p	void *	In

Type information for Class OMUCollection

Type name: defaultValues

Description: constants
Kind: Enumeration

EnumerationLiteral information for Type defaultValues

Name	Value
DefaultStartSize	20
DefaultFactor	2

Generalization information for Class OMUCollection

Generalization name: OMUAbstractContainer

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMUAbstractContainer	OMUAbstractContainer	OMUCollection

Class name: OMUAbstractContainer

Description:Abstract type less container, provides unified interface for iterations.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMUAbstractContainer

Operation name: getCurrent

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: protected
Signature: getCurrent(void * pos)
Return Type: void *
Description: Get the item in the provided position (supplied by the iterator)

Argument information for Operation getCurrent

Name	Type	Direction
pos	void *	In

Operation name: getFirst

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: protected
Signature: getFirst(void * pos)
Return Type: void
Description: iteration interface - get the initial position

Argument information for Operation getFirst

Name	Type	Direction
pos	void *	InOut

Operation name: getNext

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: protected
Signature: getNext(void * pos)
Return Type: void
Description: Get the next position for the iterator

Argument information for Operation getNext

Name	Type	Direction
pos	void *	InOut

Operation name: ~OMUAbstractContainer

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMUAbstractContainer()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Type information for Class OMUAbstractContainer

Type name: AbstractContainerPtr

Description: Abstract container pointer type
Kind: Typedef
Basic Type: OMUAbstractContainer
Multiplicity: 1
Constant: false
Reference: true
Ordered: false

Class name: OMUIterator

Description:Iterator over type less containers that derived from OMUAbstractContainer
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMUIterator

Attribute Name: thePos

Default Value: NULL
Static: false
Visibility: private
Type: void *
Stereotype:
Description: the current position of the iterator

Operation information for Class: OMUIterator

Operation name: _advance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: _advance()
Return Type: void
Description: move to the next position

Operation name: increment

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: increment()
Return Type: OMUIterator
Description: move to the next position, and return it

Operation name: OMUIterator

Initializer: thePos(iter.thePos), theLink(iter.theLink)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUIterator(OMUIterator iter)
Return Type:
Description: copy constructor

Argument information for Operation OMUIterator

Name	Type	Direction
iter	OMUIterator	In

Operation name: OMUIterator

Initializer: theLink(&l)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUIterator(OMUAbstractContainer l)
Return Type:
Description: Initialize an iterator

Argument information for Operation OMUIterator

Name	Type	Direction
l	OMUAbstractContainer	In

Operation name: OMUIterator

Initializer: theLink(l)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUIterator(AbstractContainerPtr l)
Return Type:
Description: Initialize an iterator

Argument information for Operation OMUIterator

Name	Type	Direction
l	AbstractContainerPtr	In

Operation name: OMUIterator

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMUIterator()
Return Type:
Description: default constructor

Operation name: operator*

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator*()
Return Type: void *
Description: return the current value

Operation name: operator++

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator++()
Return Type: OMUIterator
Description: operator ++ (prefix)

Operation name: operator++

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator++(int /**/)
Return Type: OMUIterator
Description: operator ++ (postfix)

Argument information for Operation operator++

Name	Type	Direction
/**/	int	In

Operation name: operator=

Initializer:
Const: false

Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(OMUIterator iter)
Return Type: OMUIterator
Description: Assignment operator

Argument information for Operation operator=

Name	Type	Direction
iter	OMUIterator	In

Operation name: reset

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: reset()
Return Type: void
Description: reset the iterator, to the container's first position

Operation name: reset

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: reset(OMUAbstractContainer newLink)
Return Type: void
Description: reset the iterator to a new container

Argument information for Operation reset

Name	Type	Direction
newLink	OMUAbstractContainer	In

Operation name: value

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: value()
Return Type: void *
Description: return the current value

Operation name: ~OMUIterator

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMUIterator()
Return Type:
Description: cleanup

Relation information for Class OMUIterator

Relation name: theLink

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theLink
LinkName:
RoleName: theLink
Type: Association
Description:

Name	Inverse	Source	Target
theLink		OMUIterator	OMUAbstractContainer

Package: Events

Type information for Package Events

Type name: OMEventNullId

Description: Null event pre-2004 id
Kind: Language
Declaration: #define %s OMNullEventId

Type name: OMEventTimeoutId

Description: Timeout event pre-2004 id
Kind: Language
Declaration: #define %s OMTimeoutEventId

Event information for Package Events

Event name: OMStartBehaviorEvent

Signature: OMStartBehaviorEvent()
Description: Support consumption of NULL transitions that follows the default transition on the appropriate active context

Event name: OMEndThreadEvent

Signature: OMEndThreadEvent()
Description: Active classes event loop termination

Event name: OMNullEvent

Signature: OMNullEvent()
Description: Null event is used for consumption of null transitions

Event name: OMCloseHandleEvent

Signature: OMCloseHandleEvent(void * handle)
Description: An event used for RTOS threads cleanup in adapters that uses the OMHandleCloser

Argument information for Event OMCloseHandleEvent

Name	Type	Direction
handle	void *	In

Event name: OMReactiveTerminationEvent

Signature: OMReactiveTerminationEvent()
Description: OMReactive graceful termination event

Attribute information for Package Events

Attribute name: OMAnyEventId

Type: ID
Stereotype:
Declaration:
Default Value: -4
Description: Any event id is used by the timer manager in extensive timeout management mode to locate timeouts that needs to be canceled.

Attribute name: OMEndThreadEventId

Type: ID
Stereotype:
Declaration:
Default Value: -6
Description: OMEndThreadEvent event id

Attribute name: OMNullEventId

Type: ID
Stereotype:
Declaration:
Default Value: -1
Description: OMNullEvent event id

Attribute name: OMStartBehaviorEventId

Type: ID
Stereotype:

Declaration:
Default Value: -5
Description: OMStartBehaviorEvent event id

Attribute name: OMTimeoutEventId

Type: ID
Stereotype:
Declaration:
Default Value: -2
Description: Timeout event id

Attribute name: OMCloseHandleEventId

Type: ID
Stereotype:
Declaration:
Default Value: -7
Description: OMCloseHandleEvent event id

Attribute name: OMCancelledEventId

Type: ID
Stereotype:
Declaration:
Default Value: -3
Description: Canceled event special id
Required for event canceling

Attribute name: OMTimeoutDelayId

Type: ID
Stereotype:
Declaration:
Default Value: -8
Description: OMDelay timeout id

Attribute name: OMReactiveTerminationEventId

Type: ID
Stereotype:
Declaration:
Default Value: -9
Description: The id of the reactive termination event

Attribute name: OMAnimWakeupEventId

Type: ID
Stereotype:
Declaration:
Default Value: -10
Description: Animation wakeup event id
Used by the thread manager to implement the wakeup API

Package: Guards

Class Information for Package: Guards

Class name: OMResourceGuard

Description:Enter-Exit object for definition of a critical section lock/unlock (the lock is done on the constructor and the unlock on the destructor)
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMResourceGuard

Attribute Name: guard

Declaration: const GUARD_TYPE&
Default Value:
Static: false
Visibility: public
Type:
Stereotype:
Description: The guard object
Note that the guard lock() & unlock() must be const

Operation information for Class: OMResourceGuard

Operation name: ~OMResourceGuard

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMResourceGuard()
Return Type:
Description: Cleanup

Operation name: OMResourceGuard

Initializer: guard(theGuard)
Const: false
Trigger: false
Abstract: false

Static: false
Virtual: false
Visibility: public
Signature: OMResourceGuard(const GUARD_TYPE& theGuard,bool aomArg(instrument))
Return Type:
Description: create a resource guard with the specified guard

Argument information for Operation OMResourceGuard

Name	Type	Direction
theGuard	const GUARD_TYPE&	In
aomArg	bool %s(instrument)	In

Operation name: OMResourceGuard

Initializer: guard(other.guard)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: OMResourceGuard(const OMResourceGuard<GUARD_TYPE>& other)
Return Type:
Description: explicitly disable copy CTOR

Argument information for Operation OMResourceGuard

Name	Type	Direction
other	const OMResourceGuard<GUARD_TYPE>&	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: operator=(const OMResourceGuard<GUARD_TYPE>& /**/)
Return Type: OMResourceGuard<GUARD_TYPE>&
Description: disable assignment operator

Argument information for Operation operator=

Name	Type	Direction
/**/	const OMResourceGuard<GUARD_TYPE>&	In

Class name: OMProtected

Description:A monitor class that uses OMOSMutex as the internal mutex object
Active: false

Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMProtected

Operation name: ~OMProtected

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMProtected()
Return Type:
Description: Cleanup

Operation name: free

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: free()
Return Type: void
Description: backward compatibility support for non OSE applications

Operation name: getGuard

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getGuard()
Return Type: OMProtected
Description: get the guard object - to allow embedding of OMProtected in OMThread

Operation name: initializeMutex

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: initializeMutex()
Return Type: void
Description: Initialize the RTOS mutex

Operation name: lock

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: lock()
Return Type: void
Description: Lock the protected object mutex.

Operation name: OMProtected

Initializer: theMutex(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMProtected(bool createMutex)
Return Type:
Description: Initialize the object with control over the initialization of the RTOS mutex

Argument information for Operation OMProtected

Name	Type	Direction
createMutex	bool	In

Operation name: OMProtected

Initializer: theMutex(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMProtected()
Return Type:
Description: Initialize the object and the RTOS mutex

Operation name: unlock

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: unlock()
Return Type: void
Description: Unlock the mutex

Operation name: cleanupMutex

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cleanupMutex()
Return Type: void
Description: Destroy the RTOS mutex

Relation information for Class OMProtected

Relation name: theMutex

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theMutex
LinkName:
RoleName: theMutex
Type: Composition
Description: The RTOS mutex

Name	Inverse	Source	Target
theMutex		OMProtected	OMOSMutex

Class name: OMGuard

Description:Enter-exit guard on OMProtected classes
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Package Information

Description: Generic guard classes

Package: DllGuards

Class Information for Package: DllGuards

Class name: OXFRefLock

Description:enter-exit guard for OXFRefManager
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Class name: OXFRefManager

Description:This class is used to maintain the references for OXF::init(..) call in DLL version of framework
Active: false
Behavior Overridden: false

Composite: false
Reactive: false

Attribute Information for Class: OXFRefManager

Attribute Name: totalReferences

Default Value: 0
Static: false
Visibility: public
Type: long
Stereotype:
Description: reference count to the number of calls

Attribute Name: oxfStarted

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: indicate if the OXF default active class is running (OXF::start() was called)

Operation information for Class: OXFRefManager

Operation name: OXFRefManager

Initializer: theMutex(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OXFRefManager()
Return Type:
Description: Initialize

Operation name: ~OXFRefManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OXFRefManager()
Return Type:
Description: Cleanup

Operation name: Increment

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false

Visibility: public
Signature: Increment()
Return Type: long
Description: Increase the count of the framework users

Operation name: Decrement

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Decrement()
Return Type: long
Description: Reduce the count of the framework users

Operation name: lock

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: lock()
Return Type: void
Description: Lock the mutex

Operation name: unlock

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: unlock()
Return Type: void
Description: Unlock the mutex

Relation information for Class OXFRefManager

Relation name: theMutex

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theMutex
LinkName:
RoleName: theMutex
Type: Composition
Description: The RTOS mutex

Name	Inverse	Source	Target

theMutex		OXFRefManager	OMOSMutex

Object information for Package DllGuards

Object name: theRefManager

Of Type: OXFRefManager
Multiplicity: 1
Description: global instance to track the number of framework users

Attribute Information for Object: OXFRefManager

Attribute Name: totalReferences

Declaration:
Default Value: 0
Static: false
Visibility: public
Type: long
Stereotype:
Description: reference count to the number of calls

Attribute Name: oxfStarted

Declaration:
Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: indicate if the OXF default active class is running (OXF::start() was called)

Operation information for Object: OXFRefManager

Operation name: OXFRefManager

Initializer: theMutex(NULL)
Const: false
Trigger: false
Body: theMutex = OMOSFactory::instance()->createOMOSMutex();
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OXFRefManager()
Return Type Name:
Description: Initialize

Operation name: ~OXFRefManager

Initializer:
Const: false
Trigger: false
Body: delete theMutex;
theMutex = NULL;
Abstract: false

Static: false
Virtual: false
Visibility: public
Signature: ~OXFRefManager()
Return Type Name:
Description: Cleanup

Operation name: Increment

Initializer:
Const: false
Trigger: false
Body: return ++totalReferences;
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Increment()
Return Type Name: long
Description: Increase the count of the framework users

Operation name: Decrement

Initializer:
Const: false
Trigger: false
Body: if (totalReferences > 0) {
	--totalReferences;
}
return totalReferences;
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Decrement()
Return Type Name: long
Description: Reduce the count of the framework users

Operation name: lock

Initializer:
Const: true
Trigger: false
Body: if (theMutex) {
	theMutex->lock();
}
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: lock()
Return Type Name: void
Description: Lock the mutex

Operation name: unlock

Initializer:
Const: true
Trigger: false

Body: if (theMutex) {
	theMutex->unlock();
}
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: unlock()
Return Type Name: void
Description: Unlock the mutex

Relation information for Object: OXFRefManager

Relation name: theMutex

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theMutex
LinkName:
RoleName: theMutex
Type: Composition
Description: The RTOS mutex

Name	Inverse	Source	Target
theMutex		OXFRefManager	OMOSMutex

Package: IO

Class Information for Package: IO

Class name: OMNotifier

Description:This class is responsible for sending of notifications to the user
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMNotifier

Operation name: notifyToError

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyToError(char* msg)
Return Type: void
Description: Sent a message to the error output

Argument information for Operation notifyToError

Name	Type	Direction
msg	char*	In

Operation name: notifyToOutput

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: notifyToOutput(char* msg)
Return Type: void
Description: Sent a message to the standard output

Argument information for Operation notifyToOutput

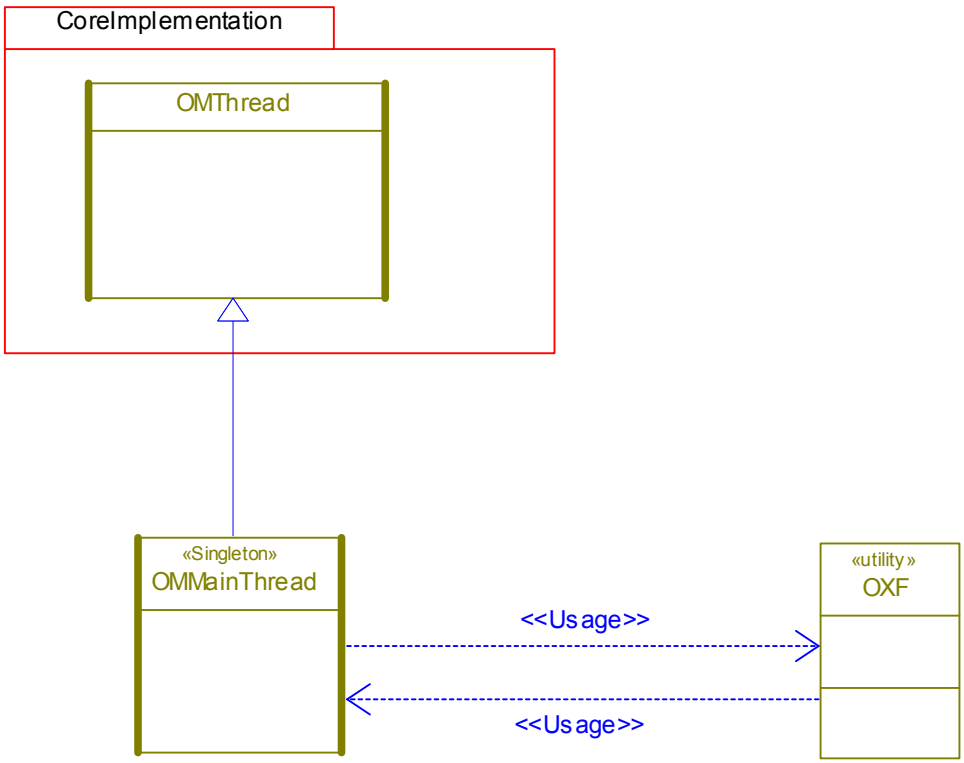
Name	Type	Direction
msg	char*	In

Package: Initialization

Object Model Diagram Information

Object Model Diagram name: Initialization classes

Description: Overview of the initialization classes



Class Information for Package: Initialization

Class name: OMMainThread

Description:The default active class for running the application main event loop.
Active: true
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMMainThread

Operation name: ~OMMainThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMMainThread()
Return Type:
Description: Cleanup

Operation name: OMMainThread

Initializer: OMThread(true) /* create a wrapper thread */
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: OMMainThread()
Return Type:
Description: force singleton

Operation name: destroyThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: destroyThread()
Return Type: void
Description: override destroyThread(), to disable deletion of statically allocated instance
call the cleanupThread() to perform cleanup

Operation name: getInstance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true

Virtual: false
Visibility: private
Signature: getInstance()
Return Type: OMMainThread
Description: actually get the main thread instance

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance(bool create)
Return Type: OMThread
Description: Get the 'default active class' singleton instance.

Argument information for Operation instance

Name	Type	Direction
create	bool	In

Operation name: startDispatching

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: startDispatching(bool doFork)
Return Type: void
Description: Start the thread & the event loop.

Argument information for Operation startDispatching

Name	Type	Direction
doFork	bool	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: operator=(OMMainThread other)
Return Type: OMMainThread
Description: disable assignment operator

Argument information for Operation operator=

Name	Type	Direction

other	OMMainThread	In

Operation name: OMMainThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: OMMainThread(OMMainThread other)
Return Type:
Description: disable copy constructor

Argument information for Operation OMMainThread

Name	Type	Direction
other	OMMainThread	In

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance(int create)
Return Type: OMThread
Description: Get the 'default active class' singleton instance.
This operation is deprecated, instance(bool) should be used instead.

Argument information for Operation instance

Name	Type	Direction
create	int	In

Generalization information for Class OMMainThread

Generalization name: OMThread

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMThread	OMThread	OMMainThread

Class name: OXF

Description:The framework entrypoint.
Provides initialization, startup and termination services.
Active: false

Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OXF

Attribute Name: rhp5CompatibleAPI

Default Value: false
Static: true
Visibility: public
Type: bool
Stereotype:
Description: This flag specifies that the framework should call methods using the Rhapsody 5.X implementation signatures to support customization of the Core implementation classes.

Attribute Name: managedTimeoutCanceling

Default Value: false
Static: true
Visibility: public
Type: bool
Stereotype:
Description: When this flag is true, the framework is responsible for timeout cancellation (via the OMTimerManager) otherwise each reactive instance is responsible for its own timeouts cancellation.

Operation information for Class: OXF

Operation name: start

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: start(bool doFork)
Return Type: void
Description: Start the framework default event loop

Argument information for Operation start

Name	Type	Direction
doFork	bool	In

Operation name: animDeregisterForeignThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: animDeregisterForeignThread(void * aomArg(theHandle))
Return Type: void

Description: Design level debugging support - de register a RTOS thread that is not associated with an IOxfActive object.

Argument information for Operation animDeregisterForeignThread

Name	Type	Direction
aomArg(theHandle)	void *	In

Operation name: animRegisterForeignThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: animRegisterForeignThread(char* aomArg(name),void * aomArg(theHandle))
Return Type: void
Description: Design level debugging support - register a RTOS thread that is not associated with an IOxfActive object

Argument information for Operation animRegisterForeignThread

Name	Type	Direction
aomArg(name)	char*	In
aomArg(theHandle)	void *	In

Operation name: delay

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: delay(OxfTimeUnit t)
Return Type: void
Description: Delay the calling thread for "t" units (currently milliseconds)

Argument information for Operation delay

Name	Type	Direction
t	OxfTimeUnit	In

Operation name: end

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: end()
Return Type: void

Description: Cleanup the framework objects

Operation name: init

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: init(int numProgArgs,char* progArgs,unsigned defaultPort,const char * defaultHost,OxfTimeUnit ticktime,unsigned maxTM,bool isRealTimeModel)
Return Type: bool
Description: initialize the framework in compatibility mode (timer, main thread, animation and etc.)

Argument information for Operation init

Name	Type	Direction
numProgArgs	int	In
progArgs	char*	In
defaultPort	unsigned %s	In
defaultHost	const char * %s	In
ticktime	OxfTimeUnit	In
maxTM	unsigned %s	In
isRealTimeModel	bool	In

Operation name: setMemoryManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: setMemoryManager(IOxfMemoryAllocator memoryManager)
Return Type: bool
Description: getter & setter for the framework memory manager
set the framework memory manager, allowed only before any memory allocation request was made

Argument information for Operation setMemoryManager

Name	Type	Direction
memoryManager	IOxfMemoryAllocator	In

Operation name: setTheDefaultActiveClass

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: setTheDefaultActiveClass(IOxfActive t)
Return Type: bool

Description: setting of the default active class is allowed only before OXF::init() is called.

Argument information for Operation setTheDefaultActiveClass

Name	Type	Direction
t	IOxfActive	In

Operation name: setTheTickTimerFactory

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: setTheTickTimerFactory(IOxfTickTimerFactory theFactory)
Return Type: bool
Description: setting the low level timers factory is allowed only before OXF::init() is called.
it is allowed to be set only once of the entire life-time of the application

Argument information for Operation setTheTickTimerFactory

Name	Type	Direction
theFactory	IOxfTickTimerFactory	In

Operation name: supportExplicitReactiveDeletion

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: supportExplicitReactiveDeletion()
Return Type: void
Description: Activate the global system support in explicit reactive instances deletion (delete <instance> instead of <instance>->terminate())

Operation name: initialize

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: initialize(int aomArg(numProgArgs),char* aomArg(progArgs),unsigned aomArg(defaultPort),const char * aomArg(defaultHost),OxfTimeUnit ticktime,unsigned maxTM,bool isRealTimeModel)
Return Type: bool
Description: initialize the framework (timer, main thread, animation and etc.)

Argument information for Operation initialize

Name	Type	Direction

aomArg(numProgArgs)	int	In
aomArg(progArgs)	char*	In
aomArg(defaultPort)	unsigned %s	In
aomArg(defaultHost)	const char * %s	In
ticktime	OxfTimeUnit	In
maxTM	unsigned %s	In
isRealTimeModel	bool	In

Relation information for Class OXF

Relation name: theTickTimerFactory

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theTickTimerFactory
LinkName:
RoleName: theTickTimerFactory
Type: Association
Description: A low-level timer factory -
if the user set the factory, these timers will be used instead of the OSAL timers

Relation name: theMemoryManager

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theMemoryManager
LinkName:
RoleName: theMemoryManager
Type: Association
Description: The framework memory manager

Relation name: theDefaultActiveClass

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theDefaultActiveClass
LinkName:
RoleName: theDefaultActiveClass
Type: Association
Description: The framework default active class (the default event loop)

Name	Inverse	Source	Target
theTickTimerFactory		OXF	IOxfTickTimerFactory
theMemoryManager		OXF	IOxfMemoryAllocator
theDefaultActiveClass		OXF	IOxfActive

Package: Macros

Package Information

Description: Various macros used by the framework and its clients

Package: CompatibilityMacros

Package Information

Description: Backward compatibility macros

Package: CodeGen50

Type information for Package CodeGen50

Type name: OMUMapItem

Description: The pre-2004 OMUMap node type

Kind: Language
Declaration: #define %s OMUMap::Item
//

Type name: OMUListItem

Description: The pre-2004 OMUList node type
Kind: Language
Declaration: #define %s OMUList::Item
//

Package: Obsolete

Type information for Package Obsolete

Type name: AnyEvent_id

Description: compatibility name
Kind: Language
Declaration: #define %s OMEventAnyEventId

Type name: CancelledEvent_id

Description: compatibility name
Kind: Language
Declaration: #define %s OMEventCancelledEventId

Type name: containersNullBlock

Description: compatibility name
Kind: Language
Declaration: #ifdef UseNullBlockContainter
#define %s OMContainersNullBlock
#endif //UseNullBlockContainter

Type name: destructiveString2X

Description: Backward compatibility for the global operation

Kind: Language
Declaration: #define %s OMDestructiveString2X

Type name: isCurrentEvent

Description: backward compatibility for the global operation isCurrentEvent(short, OMReactive*)
Kind: Language
Declaration: #define %s(id, c) ((c == 0) ? false : c->IsCurrentEvent(Id))

Type name: NOTIFY_TO_ERROR

Description: Backward compatibility for the NOTIFY_TO_ERROR macro
Kind: Language
Declaration: #define %s(msg) OM_NOTIFY_TO_ERROR(msg)

Type name: NOTIFY_TO_OUTPUT

Description: Backward compatibility for the NOTIFY_TO_OUTPUT macro
Kind: Language
Declaration: #define %s(msg) OM_NOTIFY_TO_OUTPUT(msg)

Type name: NotifyToError

Description: Backward compatibility for the global operation NotifyToError(const char*)
Kind: Language
Declaration: #define %s(msg) OMNotifier::notifyToError(msg)

Type name: NotifyToOutput

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s(msg) OMNotifier::notifyToOutput(msg)

Type name: Null_id

Description: compatibility name
Kind: Language
Declaration: #define %s OMEventNullId

Type name: OMAbstructContainer

Description: compatibility name
Kind: Language
Declaration: #define %s OMAbstractContainer

Type name: OMNotifyToError

Description: Backward compatibility for the global operation OMNotifyToError(const char*)
Kind: Language
Declaration: #define %s(msg) OMNotifier::notifyToError(msg)

Type name: OMNotifyToOutput

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s(msg) OMNotifier::notifyToOutput(msg)

Type name: OMStartBehavior_id

Description: compatibility name
Kind: Language
Declaration: #define %s OMEventStartBehaviorId

Type name: OMThreadTimer

Description: compatibility name
Kind: Typedef
Basic Type: OMTimerManager
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: OMUAbstructContainer

Description: compatibility name
Kind: Typedef
Basic Type: OMUAbstractContainer
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: OSEndApplication

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s(errorCode) OMOS::endApplication(errorCode)

Type name: OSOXFEndProlog

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s() OMOS::endProlog()

Type name: OSOXFInitEpilog

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s() OMOS::initEpilog()

Type name: OXFDelay

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s(t) OXF::delay(t)

Type name: OXFEnd

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s OXF::end

Type name: OXFEndEvent_id

Description: OMEventOXFEndEventId compatibility name
Kind: Language
Declaration: #define %s OMEventOXFEndEventId

Type name: OXFInit

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s OXF::init

Type name: OXFStart

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s OXF::start

Type name: strcmpNoCase

Description: Backward compatibility for the global operation
Kind: Language
Declaration: #define %s OMStrcmpNoCase

Type name: theOSFactory

Description: Backward compatibility for the global operation theOSFactory()
Kind: Language
Declaration: #define %s() OMOSFactory::instance()

Type name: theSysTimer

Description: the timer manager singleton
Kind: Language
Declaration: #define %s OMThreadTimer::instance()

Type name: Timeout_id

Description: Compatibility name
Kind: Language
Declaration: #define %s OMEventTimeoutId

Package: DllMacros

Type information for Package DllMacros

Type name: FRAMEWORK_DLL

Description: The framework DLL export macros
Kind: Language
Declaration: #ifdef OM_ENABLE_DLL
#ifndef RP_FRAMEWORK_DLL
#ifndef FRAMEWORK_DLL

#define RP_FRAMEWORK_DLL
#define RP_FRAMEWORK_DLLV(type) type

#else // !FRAMEWORK_DLL

#define RP_FRAMEWORK_DLLV(type) RP_FRAMEWORK_DLL

#ifdef RP_FRAMEWORK_EXPORTS
#define RP_FRAMEWORK_DLL __declspec(dllexport)
#else // !RP_FRAMEWORK_EXPORTS
#define RP_FRAMEWORK_DLL __declspec(dllimport)

#endif // RP_FRAMEWORK_EXPORTS
#endif // FRAMEWORK_DLL
#endif // RP_FRAMEWORK_DLL

```
#else // !OM_ENABLE_DLL
#define RP_FRAMEWORK_DLL
#define RP_FRAMEWORK_DLLV(type) type
#endif // OM_ENABLE_DLL
```

Package: GenMacros

Type information for Package GenMacros

Type name: GEN

Description: Send an event
Kind: Language
Declaration: #define %s(event) send(new event, OMOSEventGenerationParams(false))

Type name: GEN_BY_GUI

Description: Design level debugging - send an event via a UI that uses the framework.
Kind: Language
Declaration: #define %s(event) send(new event, OMOSEventGenerationParams((void*)OMGui))

Type name: GEN_BY_X

Description: Design level debugging - send an event from a specific context
Kind: Language
Declaration: #define %s(event,sender) send(new event, OMOSEventGenerationParams(sender))

Type name: GEN_ISR

Description: Send an event from the context of an ISR handler
Kind: Language
Declaration: #define %s(event) send(event, OMOSEventGenerationParams(true))

Package: GuardMacros

Type information for Package GuardMacros

Type name: END_REACTIVE_GUARDED_SECTION

Description: Unlock a reactive object guard
Kind: Language
Declaration: #define %s \
```
{                                                       \
        if (getThread() != NULL)  \
        {                                               \
                getThread()->unlock();      \
        }                                               \
}
```

Type name: END_THREAD_GUARDED_SECTION

Description: Unlock an active object guard
Kind: Language
Declaration: #define %s { unlock(); }

Type name: GUARD_OPERATION

Description: Guard an operation using an OMGuard enter-exit object

Kind: Language
Declaration: #define %s OMGuard _omGuard(getGuard());

Type name: OMDECLARE_GUARDED

Description: Declare an OMProtected object and helpers to support guarding of a resource
Kind: Language
Declaration: #define %s
```
                                \
public:
                                                \
        inline void lock() const { m_omGuard.lock(); }
        \
        inline void unlock() const { m_omGuard.unlock(); }
        \
        inline const OMProtected& getGuard() const { return m_omGuard; }    \
private:
                                        \
        OMProtected m_omGuard;
```

Type name: START_DTOR_REACTIVE_GUARDED_SECTION

Description: Guard a reactive object destructor
Kind: Language
Declaration: #define %s
```
                                \
{
                                \
        if (!isUnderDestruction())                                                      \
        {
                                \
                setUnderDestruction();
        \
                if (getToGuardReactive() && (getThread() != NULL))          \
                {
                        \
                        getThread()->lock();
        \
                }
                        \
        }
                        \
}
```

Type name: START_DTOR_THREAD_GUARDED_SECTION

Description: Guard an active object destructor
Kind: Language
Declaration: #define %s
```
                                                                                \
{                                                                               \
        if (!isUnderDestruction())              \
        {                                                                       \
                setUnderDestruction();                  \
                if (shouldGuardThread())  \
                {                                                               \
                        OMThread::lock();                       \
                }                                                               \
        }                                                                       \
```

```
}
```

Type name: START_REACTIVE_GUARDED_SECTION

Description: Lock a reactive object guard
Kind: Language
Declaration:
```
#define %s                                                      \
{                                                               \
        if (getThread() != NULL)  \
        {                                                       \
                getThread()->lock();         \
        }                                                       \
}
```

Type name: START_THREAD_GUARDED_SECTION

Description: Lock an active object guard
Kind: Language
Declaration:
```
#define %s { lock(); }
```

Package: MemoryManagerMacros

Type information for Package MemoryManagerMacros

Type name: REPLACEMENT_NEW

Description: Support OS without global placement new/delete operators
Kind: Language
Declaration:
```
#if (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) &&
!defined(OMOMATE))
#ifdef OM_NO_OS_REPLACEMENT_NEW
inline void * operator new(size_t, void * ptr) { return ptr; }
inline void operator delete(void *, void *) {return; }
#else
#ifdef OM_STL
#include <new>
#else
#include <new.h>
#endif // OM_STL
#endif // OM_NO_OS_REPLACEMENT_NEW
#endif // (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) && !defined(OMOMATE))
```

Type name: OS_ASSERT

Description: Support OS without support in ANSI assert()
Kind: Language
Declaration:
```
#if (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) &&
!defined(OMOMATE))
#ifdef OM_DISABLE_ASSERT
// Support disabling of assert().
// Done to avoid LINT warnings due to Microsoft implementation.
#ifdef assert
#undef assert
#endif // assert
#define assert(condition)
#else // !OM_DISABLE_ASSERT
#ifdef OM_NO_OS_ASSERT
```

```
#ifndef assert
#define assert(condition) if (!(condition)) exit(-1);
#endif // assert
#else
#include <assert.h>
#endif // OM_NO_OS_ASSERT
#endif // OM_DISABLE_ASSERT
#endif // (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) && !defined(OMOMATE))
```

Type name: NEW_DUMMY_PARAM

Description: Add dummy size parameter to the new operator - compiler dependent code
Kind: Language
Declaration:
```
#ifdef OM_NEW_OPERATOR_NEEDS_DUMMY_PARAM
#define %s ,size_t = 0
#else
#define %s
#endif // OM_NEW_OPERATOR_NEEDS_DUMMY_PARAM
```

Type name: OMNEW

Description: Request memory for a given number of instances of the given type
Kind: Language
Declaration:
```
#if (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) &&
!defined(OMOMATE))
#define %s(type, size) new(OMMemoryManager::getMemoryManager()->getMemory(sizeof(type) *
(size_t)(size))) type[(size_t)(size)]
#else
// use the global new operator
#define %s(type, size) new type[(size_t)(size)]
#endif // (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) && !defined(OMOMATE))
```

Type name: OMDELETE

Description: Return memory allocated by OMNEW
Kind: Language
Declaration:
```
#if (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) &&
!defined(OMOMATE))

#define %s(object,size) OMMemoryManager::getMemoryManager()-> \
        returnMemory(
                \
                reinterpret_cast<void*>(object),                                        \
                static_cast<size_t>(size))

#else
// the dummy variable is used to avoid compilation warnings,
//  it is required for the other definition of the OMDELETE() macro - when the memory manager is used.
#define OMDELETE(object, dummy)                 \
        if ((dummy) == 0) { }                   \
        delete[] (object);

#endif // (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) && !defined(OMOMATE))
```

Type name: OMGET_MEMORY

Description: Request memory for a given number of bytes
Kind: Language

Declaration: #if (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) &&
!defined(OMOMATE))
#define %s(size) OMMemoryManager::getMemoryManager()->getMemory((size_t)(size))
#else
// use 'regular' new & delete for memory allocation
#define %s(size) ::operator new((size_t) (size))
#endif // (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) && !defined(OMOMATE))

Type name: OM_DECLARE_FRAMEWORK_MEMORY_ALLOCATION_OPERATORS

Description: Override on the new and delete operators to obtain memory from the framework memory manager
Kind: Language
Declaration: #if (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) &&
!defined(OMOMATE))
#define %s \
public: \
static void* operator new (size_t size NEW_DUMMY_PARAM) \
{ \
return OMMemoryManager::getMemoryManager()->getMemory(size); \
} \
static void* operator new[] (size_t size NEW_DUMMY_PARAM) \
{ \
return OMMemoryManager::getMemoryManager()->getMemory(size); \
} \
static void operator delete (void * object, size_t size) \
{ \
if (object != NULL) \
OMMemoryManager::getMemoryManager()->returnMemory(object,size); \
} \
static void operator delete[] (void * object, size_t size) \
{ \
if (object != NULL) \
OMMemoryManager::getMemoryManager()->returnMemory(object,size); \
}

#else
// empty
#define %s
#endif // (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) && !defined(OMOMATE))

Type name: OM_MEMORY_MANAGER_SWITCH_HELPER_POOL_SIZE

Description: The pool size for the memory manager switch helper allocated memory list
Kind: Language
Declaration: #if (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) &&
!defined(OMOMATE))
#ifndef %s
#define %s 512
#endif // ifndef %s
#endif // (!defined(OM_NO_FRAMEWORK_MEMORY_MANAGER) && !defined(OMOMATE))

Package: MemoryPoolsMacros

Type information for Package MemoryPoolsMacros

Type name: DECLARE_ALLOCATION_OPERATORS

Description: The framework memory pool new/delete operators declaration.
Kind: Language
Declaration: #define %s
\
static void operator delete(void *deadObject, size_t size); \
static void * operator new(size_t size NEW_DUMMY_PARAM) OM_NO_THROW;

Type name: DECLARE_MEMORY_ALLOCATOR

Description: The framework memory pool operations that are added to the controlled classes.
Kind: Language
Declaration: #define %s(CLASSNAME,INITNUM)
\
public:
\
CLASSNAME* OMMemoryPoolNextChunk;
\
DECLARE_ALLOCATION_OPERATORS
\
static void OMMemoryPoolIsEmpty();
\
static void OMMemoryPoolSetIncrement(int value);
\
static void OMCallMemoryPoolIsEmpty(bool flagValue);
\
static void OMSetMemoryAllocator(CLASSNAME*(*newAllocator)(int)); \
static OMSelfLinkedMemoryAllocator<CLASSNAME,INITNUM>&
myOMMemoryAllocator();

Type name: IMPLEMENT_ALLOCATION_OPERATORS

Description: The framework memory pool new/delete operators implementation.
Kind: Language
Declaration: #define %s(CLASSNAME)
\
void CLASSNAME::operator delete(void *deadObject, size_t size)
\
{
\

```
        myOMMemoryAllocator().returnMemory(deadObject,size);
    \
    }
                                                                    \
    void * CLASSNAME::operator new(size_t size NEW_DUMMY_PARAM_IMP)
OM_NO_THROW         \
    {
                                                                    \
        void * mem = myOMMemoryAllocator().getMemory(size);
        \
        if (mem == NULL) {
                                                \
            MEMORY_ALLOCATION_FAIL_MSG(CLASSNAME,mem)
                                    \
        }
                                                                    \
        return mem;
                                                        \
    }
```

Type name: IMPLEMENT_MEMORY_ALLOCATOR

Description: The framework memory pool operations that are added to the controlled classes.
Kind: Language
Declaration:

```
#define %s(CLASSNAME,INITNUM,INCREMENTNUM,ISPROTECTED)
                                \
    OMSelfLinkedMemoryAllocator<CLASSNAME,INITNUM>&
CLASSNAME::myOMMemoryAllocator()                                    \
    {

    \
        static OMSelfLinkedMemoryAllocator<CLASSNAME,INITNUM>
allocator(INCREMENTNUM,ISPROTECTED);    \
        return allocator;
                                                                        \
    }

    \
    IMPLEMENT_ALLOCATION_OPERATORS(CLASSNAME)
                                                                        \
    void CLASSNAME::OMMemoryPoolSetIncrement(int value)
                                                        \
    {

    \
        myOMMemoryAllocator().setIncrementNum(value);
                                                        \
    }

    \
    void CLASSNAME::OMMemoryPoolIsEmpty()
                                                                    \
    {

    \
        POOL_REALLOCATION_MSG(CLASSNAME,INCREMENTNUM)
                                                                    \
```

```
        }

        \
        void CLASSNAME::OMCallMemoryPoolIsEmpty(bool flagValue)
                                                                \
        {

        \
                myOMMemoryAllocator().callMemoryPoolIsEmpty(flagValue);
                                                                \
        }

        \
        void CLASSNAME::OMSetMemoryAllocator(CLASSNAME*(*newAllocator)(int))
                                            \
        {

        \
                myOMMemoryAllocator().setAllocator(newAllocator);
                                                                \
        }
```

Type name: MEMORY_ALLOCATION_FAIL_MSG

Description: The framework memory pool fail message
Kind: Language
Declaration: #ifdef _OMINSTRUMENT

```
        #define  %s(CLASSNAME,MEM)                                  \
                        OMString s = #CLASSNAME ;
                \
                        s += ": Memory allocation for new instances failed.\n";          \
                        OM_NOTIFY_TO_ERROR(s);

#else
        #define  %s(CLASSNAME,MEM)
#endif // _OMINSTRUMENT
```

Type name: NEW_DUMMY_PARAM_IMP

Description: The framework memory pool dummy new parameter (required by some compilers)
Kind: Language
Declaration: #ifdef OM_NEW_OPERATOR_NEEDS_DUMMY_PARAM
#define %s ,size_t
#else // !OM_NEW_OPERATOR_NEEDS_DUMMY_PARAM
#define %s
#endif // OM_NEW_OPERATOR_NEEDS_DUMMY_PARAM

Type name: OM_DYNAMIC_POOL_INITIALIZATION_SIZE

Description: The initial pool size for dynamic initialization
Kind: Language
Declaration: #define %s 0

Type name: POOL_REALLOCATION_MSG

Description: The framework memory pool additional allocation message
Kind: Language
Declaration: #ifdef _OMINSTRUMENT

```
        #define %s(CLASSNAME,INCREMENTSIZE)
        \
                OMString s = #CLASSNAME ;
                                        \
                s += ": Memory pool is empty, allocating memory pool for additional ";\
                s += #INCREMENTSIZE ;
                                                \
                s += " instances.\n";
                                        \
                OM_NOTIFY_TO_OUTPUT(s);
#else
        #define %s(CLASSNAME,INCREMENTSIZE)
#endif // _OMINSTRUMENT
```

Type name: RESET_MEMORY_ALLOCATOR

Description: Reset the memory pool
Kind: Language
Declaration: #define %s(CLASSNAME) \
        OMMemoryPoolNextChunk = reinterpret_cast<CLASSNAME*>(0);

Package: NotifyMacros

Type information for Package NotifyMacros

Type name: OM_NOTIFY_TO_ERROR

Description: Notify an error message
Kind: Language
Declaration: #define %s(msg) OMNotifier::notifyToError(msg)

Type name: OM_NOTIFY_TO_OUTPUT

Description: Notify a message
Kind: Language
Declaration: #define %s(msg) OMNotifier::notifyToOutput(msg)

Package: PortMacros

Type information for Package PortMacros

Type name: _OPORT

Description: Get the out bound of the provided port
Kind: Language
Declaration: #define %s(p) p->getOutBound()

Type name: IS_PORT

Description: Check if the current event was sent via the specified port
Kind: Language
Declaration: #define %s(p) getCurrentEvent()->getPort() == (void *)(get_##p())

Type name: IS_PORT_AT

Description: Check if the current event was sent via the specified port when using a port with multiplicity > 1

Kind: Language
Declaration: #define %s(p,i) getCurrentEvent()->getPort() == (void *)(get_##p(i))

Type name: OPORT

Description: Get the out bound of the provided port
Kind: Language
Declaration: #define %s(p) OUT_PORT(p)

Type name: OPORT_AT

Description: Get the out bound of the provided port when using a port with multiplicity > 1
Kind: Language
Declaration: #define %s(p, i) OUT_PORT_AT(p, i)

Type name: OUT_PORT

Description: Get the out bound of the provided port
Kind: Language
Declaration: #define %s(p) _OPORT(get_##p())

Type name: OUT_PORT_AT

Description: Get the out bound of the provided port when using a port with multiplicity > 1
Kind: Language
Declaration: #define %s(p, i) _OPORT(get_##p(i))

Package: STLMacros

Type information for Package STLMacros

Type name: OM_USE_STL

Description: Use the std namespace
Kind: Language
Declaration: #ifdef %s
namespace std {}
using namespace std;
#define OM_STL
#endif // %s

Type name: STD_NAMESPACE

Description: Some compilers don't support the std:: namespace
You may omit it from the signature by compiling your application with NO_STD_NAMESPACE compiler switch
Kind: Language
Declaration: #define %s std::
#ifdef NO_STD_NAMESPACE
#undef %s
#define %s
#endif // NO_STD_NAMESPACE

Package: StatechartMacros

Type information for Package StatechartMacros

Type name: IS_COMPLETED

Description: Check if the specified state reached completion.
A macro is used to support both flat and reusable statechart implementation.
Kind: Language
Declaration: #ifndef OM_REUSABLE_STATECHART_IMPLEMENTATION
#define %s(state) state##_isCompleted()
#else
#define %s(state) state->isCompleted()
#endif // OM_REUSABLE_STATECHART_IMPLEMENTATION

Type name: IS_EVENT_TYPE_OF

Description: Test if the current event is of the type of the given id.
A macro is used to support both flat and reusable statechart implementation.
Kind: Language
Declaration: #ifndef OM_REUSABLE_STATECHART_IMPLEMENTATION
#define %s(id) ((getCurrentEvent() != NULL) ? getCurrentEvent()->isTypeOf((id)) : false)
#else
#define %s(id) (((concept != NULL) && ((concept->getCurrentEvent()) != NULL)) ? (concept->getCurrentEvent())->isTypeOf((id)) : false)
#endif // OM_REUSABLE_STATECHART_IMPLEMENTATION

Type name: IS_IN

Description: Check if the given state is active.
A macro is used to support both flat and reusable statechart implementation.
Kind: Language
Declaration: #ifndef OM_REUSABLE_STATECHART_IMPLEMENTATION
#define %s(state) state##_IN()
#else
#define %s(state) state->in()
#endif // OM_REUSABLE_STATECHART_IMPLEMENTATION

Type name: OMREPLY

Description: Set a triggered operation return value.
A macro is used to support both flat and reusable statechart implementation.
Kind: Language
Declaration: #define %s(retVal) (params->om_reply = retVal)

Type name: OMSETPARAMS

Description: Set a local variable called params of the specific event type to enable access to the current event data.
A macro is used to support both flat and reusable statechart implementation.
Kind: Language
Declaration: #define %s(type) type* params = static_cast<type*>(getCurrentEvent());

Type name: reply

Description: Shorted name for OMREPLY
Kind: Language
Declaration: #define %s(retVal) OMREPLY(retVal)

Type name: SETPARAMS

Description: Shorted name for OMSETPARAMS
Kind: Language
Declaration: #define %s(type) OMSETPARAMS(type)

Type name: OM_CURRENT_EVENT_ID

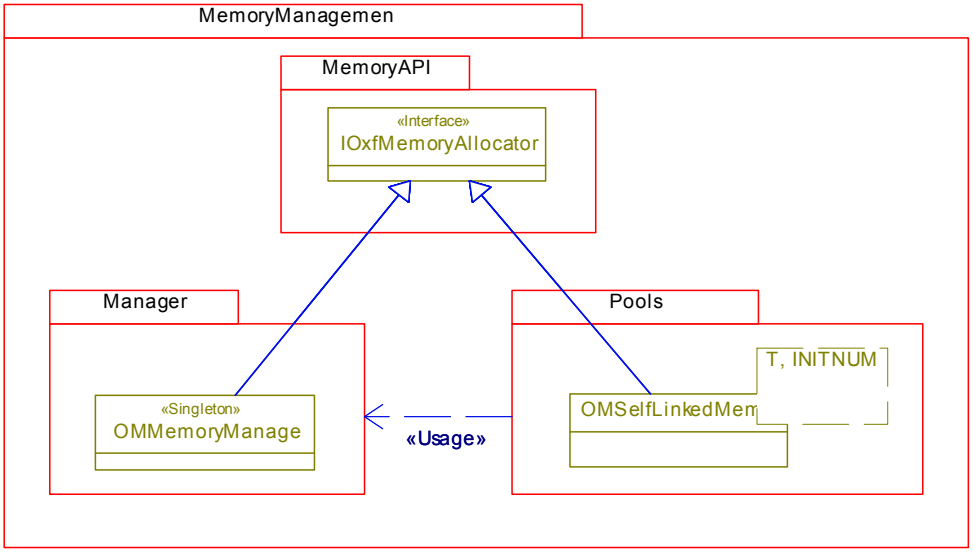
Description: Get the current event id.
A macro is used to support both flat and reusable statechart implementation.
Kind: Language
Declaration: #ifndef OM_REUSABLE_STATECHART_IMPLEMENTATION
#define %s ((getCurrentEvent() != NULL) ? getCurrentEvent()->getId() : 0)
#else
#define %s (((concept != NULL) && ((concept->getCurrentEvent()) != NULL)) ? (concept->getCurrentEvent())->getId() : 0)
#endif // OM_REUSABLE_STATECHART_IMPLEMENTATION

Package: MemoryManagement

Object Model Diagram Information

Object Model Diagram name: Memory Services

Description: Overview on the memory management services provided by the framework



Package Information

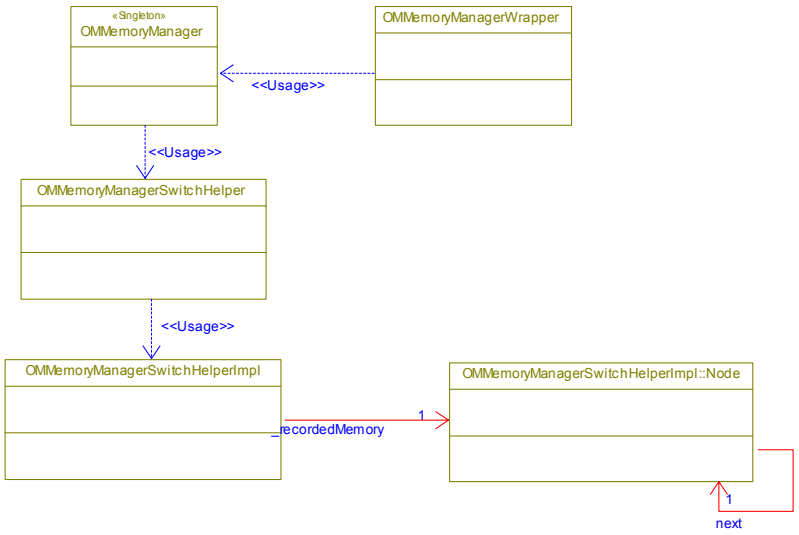
Description: Memory management support

Package: Manager

Object Model Diagram Information

Object Model Diagram name: Memory manager overview

Description: Overview on the memory manager architecture



Class Information for Package: Manager

Class name: OMMemoryManagerSwitchHelper

Description:Support switch of the memory manager after memory was already requested.
This support is required since the memory must return via the manager that allocated it.
This class supply the interface, the actual implementation is provided by the
OMMemoryManagerSwitchHelperImpl that should not be accessed directly.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMMemoryManagerSwitchHelper

Operation name: cleanup

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cleanup()
Return Type: void
Description: cleanup the allocated memory list

Operation name: findMemory

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: findMemory(void * ommmswArg(memory))
Return Type: bool

Description: search for a recorded memory allocation
return 'true' if the memory was found in the recorded memory
or 'false' when the memory is not found

Argument information for Operation findMemory

Name	Type	Direction
ommmswArg(memory)	void *	In

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance()
Return Type: OMMemoryManagerSwitchHelper
Description: return the internally used singleton instance
of the OMMemoryManagerSwitchHelper

Operation name: isLogEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isLogEmpty()
Return Type: bool
Description: check if the memory log is empty

Operation name: recordMemoryAllocation

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: recordMemoryAllocation(void * ommmswArg(memory))
Return Type: bool
Description: record a single memory allocation
return true on success

Argument information for Operation recordMemoryAllocation

Name	Type	Direction
ommmswArg(memory)	void *	In

Operation name: recordMemoryDeallocation

Initializer:

Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: recordMemoryDeallocation(void * ommmswArg(memory))
Return Type: bool
Description: record a single memory deallocation
return true if memory record found & removed ok

Argument information for Operation recordMemoryDeallocation

Name	Type	Direction
ommmswArg(memory)	void *	In

Operation name: shouldUpdate

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: shouldUpdate()
Return Type: bool
Description: Check if the switch helper should be updated

Operation name: setUpdateState

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setUpdateState(bool ommmswArg(val))
Return Type: void
Description: Set the switch helper updating state

Argument information for Operation setUpdateState

Name	Type	Direction
ommmswArg(val)	bool	In

Class name: OMMemoryManager

Description:The framework default memory manager, uses the global new/delete operators to get and return memory.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMMemoryManager

Attribute Name: _singletonDestroyed

Default Value: false
Static: true
Visibility: private
Type: bool
Stereotype:
Description: Singleton state flag, used to identify that the memory manager singleton was destroyed (while exit())

Operation information for Class: OMMemoryManager

Operation name: ~OMMemoryManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMMemoryManager()
Return Type:
Description: destructor

Operation name: getDefaultMemoryManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: getDefaultMemoryManager()
Return Type: IOxfMemoryAllocator
Description: get the default (internal) memory manager

Operation name: getMemory

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getMemory(size_t size)
Return Type: void *
Description: get memory for an instance

Argument information for Operation getMemory

Name	Type	Direction
size	size_t	In

Operation name: getMemoryManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: getMemoryManager()
Return Type: IOxfMemoryAllocator
Description: get the actual memory manager

Operation name: OMMemoryManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMMemoryManager(bool theFrameworkSingleton)
Return Type:
Description: constructor

Argument information for Operation OMMemoryManager

Name	Type	Direction
theFrameworkSingleton	bool	In

Operation name: returnMemory

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: returnMemory(void * object,size_t /* size */)
Return Type: void
Description: Return the memory of the object

Argument information for Operation returnMemory

Name	Type	Direction
object	void *	In
/* size */	size_t	In

Generalization information for Class OMMemoryManager

Generalization name: IOxfMemoryAllocator

Description:
Virtual: false
Visibility: public

Extension Point:

Name	Base	Derived
IOxfMemoryAllocator	IOxfMemoryAllocator	OMMemoryManager

Class name: OMMemoryManagerWrapper

Description:A wrapper for the memory manager used to avoid circular dependencies
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMMemoryManagerWrapper

Operation name: getMemory

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: getMemory(unsigned long size)
Return Type: void *
Description: Request memory using OMGET_MEMORY

Argument information for Operation getMemory

Name	Type	Direction
size	unsigned long	In

Operation name: Delete

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: Delete(void * obj,unsigned long size)
Return Type: void
Description: Delete an object using OMDELETE

Argument information for Operation Delete

Name	Type	Direction
obj	void *	In
size	unsigned long	In

Class name: OMMemoryManagerSwitchHelperImpl

Description:Support switch of the memory manager after memory was already requested.
This support is required since the memory must return via the manager that allocated it.
This class perform the actual recording of the memory it should no be accessed or included directly.

Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMMemoryManagerSwitchHelperImpl

Attribute Name: shouldUpdateLog

Default Value: true
Static: false
Visibility: public
Type: bool
Stereotype:
Description: should update the log state flag

Operation information for Class: OMMemoryManagerSwitchHelperImpl

Operation name: ~OMMemoryManagerSwitchHelperImpl

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMMemoryManagerSwitchHelperImpl()
Return Type:
Description: DTOR

Operation name: cleanup

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cleanup()
Return Type: void
Description: cleanup the allocated memory list

Operation name: findMemory

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: findMemory(void * memory)
Return Type: bool
Description: search for a recorded memory allocation
return 'true' if the memory was found in the recorded memory
or 'false' when the memory is not found

Argument information for Operation findMemory

Name	Type	Direction
memory	void *	In

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance()
Return Type: OMMemoryManagerSwitchHelperImpl
Description: return the internally used singleton instance
of the OMMemoryManagerSwitchHelper

Operation name: isLogEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: isLogEmpty()
Return Type: bool
Description: check if the memory log is empty

Operation name: OMMemoryManagerSwitchHelperImpl

Initializer: _recordedMemory(NULL)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMMemoryManagerSwitchHelperImpl()
Return Type:
Description: CTOR

Operation name: recordMemoryAllocation

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: recordMemoryAllocation(void * memory)
Return Type: bool
Description: record a single memory allocation

return true on success

Argument information for Operation recordMemoryAllocation

Name	Type	Direction
memory	void *	In

Operation name: recordMemoryDeallocation

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: recordMemoryDeallocation(void * memory)
Return Type: bool
Description: record a single memory deallocation
return true if memory record found & removed ok

Argument information for Operation recordMemoryDeallocation

Name	Type	Direction
memory	void *	In

Relation information for Class OMMemoryManagerSwitchHelperImpl

Relation name: _recordedMemory

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: _recordedMemory
LinkName:
RoleName: _recordedMemory
Type: Association
Description: The head of the recorded memory list

Name	Inverse	Source	Target
_recordedMemory		OMMemoryManagerSwitchHelperImpl	Node

Class Information for Class: OMMemoryManagerSwitchHelperImpl

Class name: Node

Description:A node in the memory managers list
Used by OMMemoryManagerSwitchHelper
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: Node

Attribute Name: allocatedMemory

Default Value:
Static: false
Visibility: public
Type: void *
Stereotype:
Description: The allocated memory address

Operation information for Class: Node

Operation name: Node

Initializer: OMMemoryPoolNextChunk(0), allocatedMemory(mem), next(nextNode)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Node(void * mem,Node* nextNode)
Return Type:
Description: Initialize a node

Argument information for Operation Node

Name	Type	Direction
mem	void *	In
nextNode	Node*	In

Operation name: Node

Initializer: OMMemoryPoolNextChunk(0), allocatedMemory(0), next(0)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Node()
Return Type:
Description: Initialize an empty node

Operation name: outOfMemoryAssert

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: outOfMemoryAssert(int /**/)
Return Type: Node*
Description: assert when run out of memory

Argument information for Operation outOfMemoryAssert

Name	Type	Direction
/**/	int	In

Relation information for Class Node

Relation name: next

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: next
LinkName:
RoleName: next
Type: Association
Description: The next node in the memory list

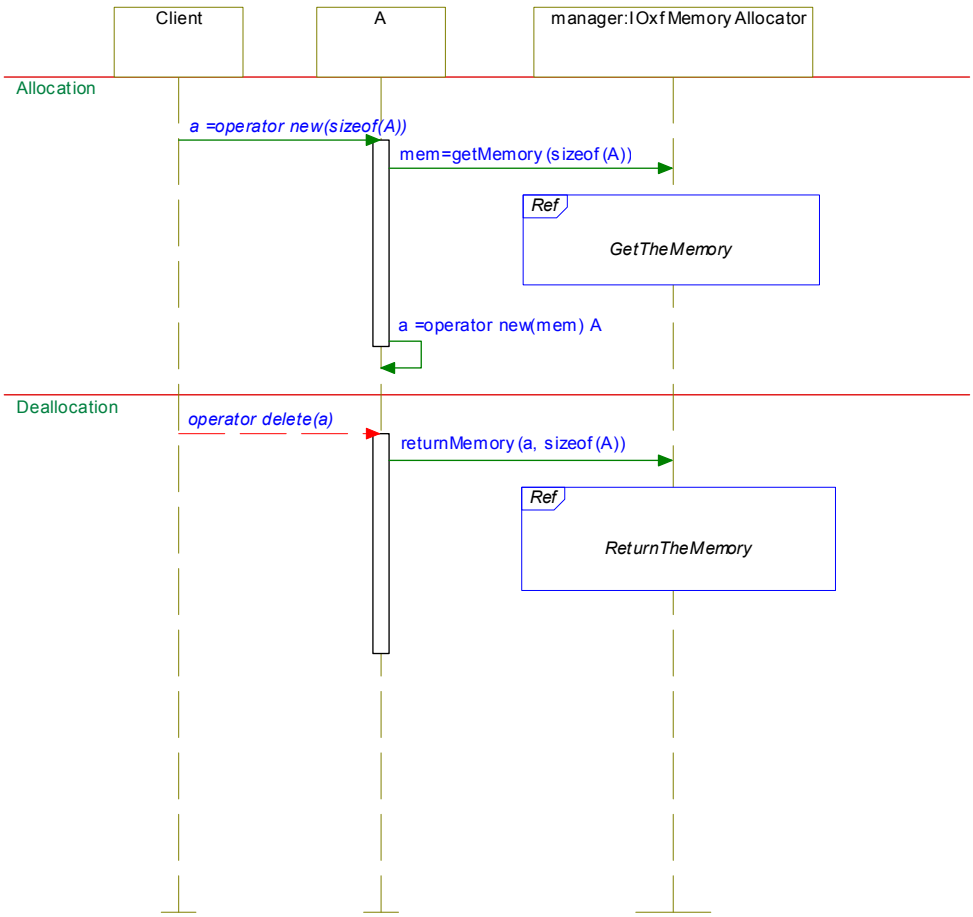
Name	Inverse	Source	Target
next		Node	Node

Package: MemoryAPI

Sequence Diagram Information

Sequence Diagram name: Usage of the memory manager

Description:



Class Information for Package: MemoryAPI

Class name: IOxfMemoryAllocator

Description:Memory manager interface
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: IOxfMemoryAllocator

Operation name: getMemory

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: true
Visibility: public
Signature: getMemory(size_t size)
Return Type: void *
Description: get a memory block of the specified size

Argument information for Operation getMemory

Name	Type	Direction
size	size_t	In

Operation name: returnMemory

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: true
Visibility: public
Signature: returnMemory(void * object,size_t size)
Return Type: void
Description: Return the memory of the object.
The size argument is available for optimization of the memory manager implementation (for example when several pools are used based on the requested memory size).

Argument information for Operation returnMemory

Name	Type	Direction
object	void *	In
size	size_t	In

Operation name: ~IOxfMemoryAllocator

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfMemoryAllocator()
Return Type:
Description: Virtual destructor to enable polymorphic deletion

Package Information

Description: Generic memory management API

Package: Compatibility

Type information for Package Compatibility

Type name: OMAbstractMemoryAllocator

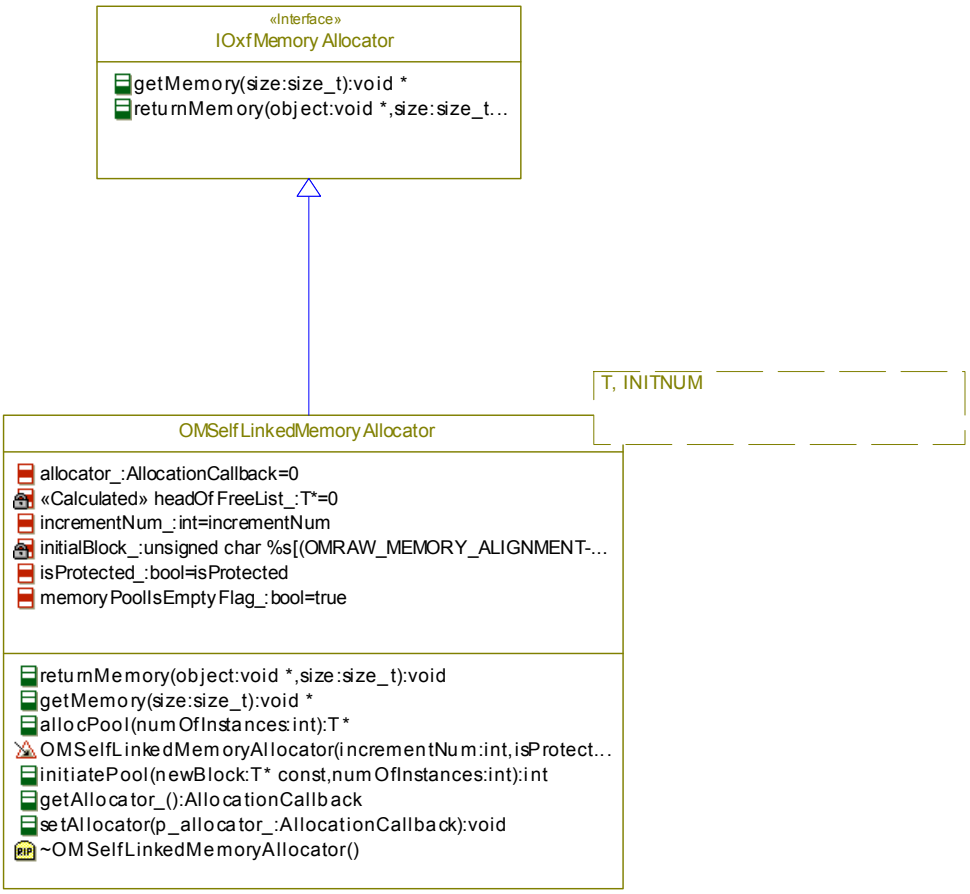
Description: Rhapsody 5.X name of the IOxfMemoryAllocator interface
Kind: Typedef
Basic Type: IOxfMemoryAllocator
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Package: Pools

Object Model Diagram Information

Object Model Diagram name: pool implementation

Description: The memory pool implementation overview



Class Information for Package: Pools

Class name: OMSelfLinkedMemoryAllocator

Description:Memory pool implementation
Active: false
Behavior Overridden: false
Composite: true
Reactive: false

Attribute Information for Class: OMSelfLinkedMemoryAllocator

Attribute Name: allocator_

Default Value: 0
Static: false
Visibility: public
Type: AllocationCallback
Stereotype:
Description: callback function to override the memory allocation scheme

Attribute Name: headOfFreeList_

Declaration: T*
Default Value: 0
Static: false
Visibility: private
Type:
Stereotype:
Description: head of the free list

Attribute Name: incrementNum_

Default Value: incrementNum
Static: false
Visibility: public
Type: int
Stereotype:
Description: how much (in instance number) to increment if initial pool is exhausted

Attribute Name: initialBlock_

Declaration: unsigned char %s[(OMRAW_MEMORY_ALIGNMENT-1)+(INITNUM*sizeof(T))]
Default Value:
Static: false
Visibility: private
Type:
Stereotype:
Description: initial pool

Attribute Name: isProtected_

Default Value: isProtected
Static: false
Visibility: public
Type: bool
Stereotype:
Description: activate the mutex

Attribute Name: memoryPoolIsEmptyFlag_

Default Value: true
Static: false
Visibility: public
Type: bool
Stereotype:
Description: when the flag is false, memoryPoolIsEmpty() will not be called

Operation information for Class: OMSelfLinkedMemoryAllocator

Operation name: returnMemory

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: returnMemory(void * object,size_t size)
Return Type: void
Description: return a memory of object of the specified size

Argument information for Operation returnMemory

Name	Type	Direction
object	void *	In
size	size_t	In

Operation name: getMemory

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getMemory(size_t size)
Return Type: void *
Description: get a memory block of the specified size

Argument information for Operation getMemory

Name	Type	Direction
size	size_t	In

Operation name: allocPool

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: allocPool(int numOfInstances)
Return Type: T*
Description: allocate memory pool big enough to hold numOfInstances instances of type T

Argument information for Operation allocPool

Name	Type	Direction
numOfInstances	int	In

Operation name: OMSelfLinkedMemoryAllocator

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMSelfLinkedMemoryAllocator(int incrementNum,bool isProtected)
Return Type:
Description: construct the allocator, specify whether it is protected
and how much additional memory should be allocated if the initial pool is exhausted

Argument information for Operation OMSelfLinkedMemoryAllocator

Name	Type	Direction
incrementNum	int	In
isProtected	bool	In

Operation name: initiatePool

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: initiatePool(T* const newBlock,int numOfInstances)
Return Type: int
Description: initiate the bookkeeping for the allocated pool

Argument information for Operation initiatePool

Name	Type	Direction
newBlock	T* const	In
numOfInstances	int	In

Operation name: lock

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: lock()
Return Type: void
Description: lock critical section

Operation name: unlock

Initializer:
Const: true
Trigger: false
Abstract: false

Static: false
Virtual: false
Visibility: private
Signature: unlock()
Return Type: void
Description: unlock critical section

Operation name: getAllocator_

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getAllocator_()
Return Type: AllocationCallback
Description: Get the memory allocation callback function

Operation name: setAllocator

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setAllocator(AllocationCallback p_allocator_)
Return Type: void
Description: Set the memory pool additional memory allocation callback

Argument information for Operation setAllocator

Name	Type	Direction
p_allocator_	AllocationCallback	In

Operation name: ~OMSelfLinkedMemoryAllocator

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMSelfLinkedMemoryAllocator()
Return Type:
Description: Cleanup

Type information for Class OMSelfLinkedMemoryAllocator

Type name: AllocationCallback

Description: Additional memory allocation callback
Kind: Language

Declaration: typedef omtypename T* (*%s)(int);

Generalization information for Class OMSelfLinkedMemoryAllocator

Generalization name: IOxfMemoryAllocator

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfMemoryAllocator	IOxfMemoryAllocator	OMSelfLinkedMemoryAllocator

Relation information for Class OMSelfLinkedMemoryAllocator

Relation name: myGuard_

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: myGuard_
LinkName:
RoleName: myGuard_
Type: Composition
Description: The pool guard

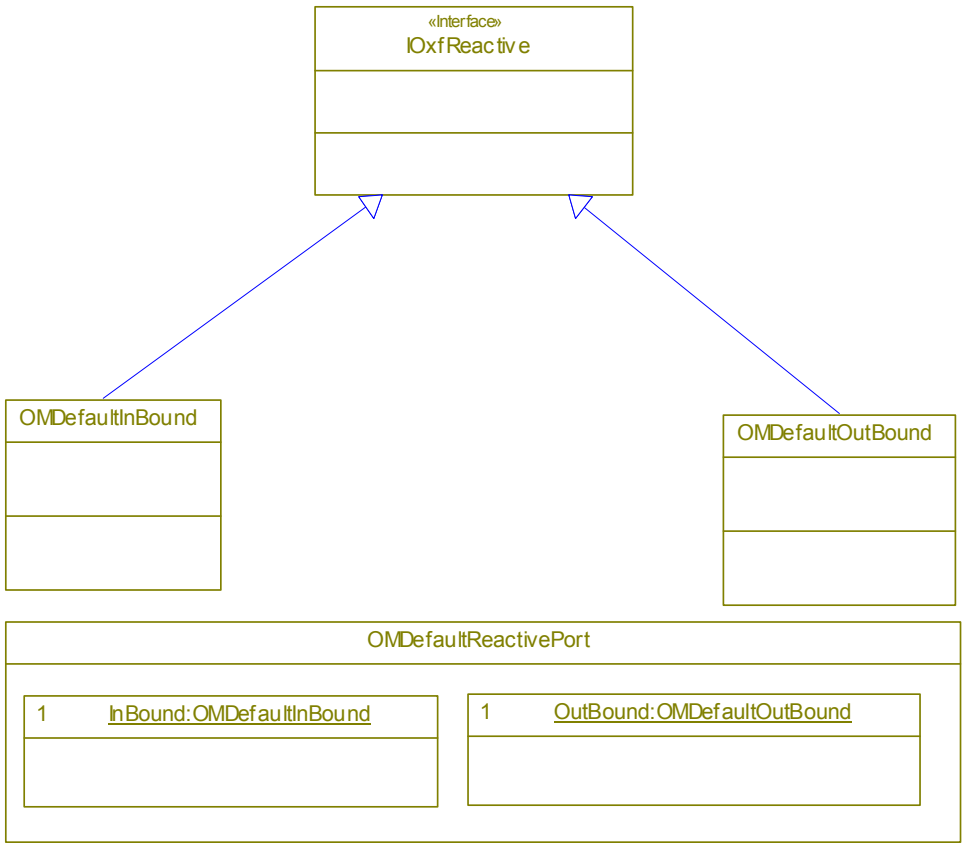
Name	Inverse	Source	Target
myGuard_		OMSelfLinkedMemoryAllocator	OMProtected

Package: Ports

Object Model Diagram Information

Object Model Diagram name: support ports default behavior

Description: The default reactive port architecture



Class Information for Package: Ports

Class name: OMDefaultInBound

Description:Default behavioral in-bound port of a reactive class implementation.
Forward the messages to the reactive state machine.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMDefaultInBound

Attribute Name: port

Default Value: 0
Static: false
Visibility: public
Type: void *
Stereotype:
Description: just for stamping incoming events

Operation information for Class: OMDefaultInBound

Operation name: handleEvent

Initializer:
Const: false

Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: handleEvent(IOxfEvent /*ev*/)
Return Type: TakeEventStatus
Description: consume an event

Argument information for Operation handleEvent

Name	Type	Direction
/*ev*/	IOxfEvent	In

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: send(IOxfEvent ev,IOxfEventGenerationParams params)
Return Type: bool
Description: send an event to the active context queue

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In
params	IOxfEventGenerationParams	In

Operation name: startBehavior

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: startBehavior()
Return Type: bool
Description: initialize the reactive instance state machine

Operation name: destroy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: destroy()
Return Type: void

Description: destroy the reactive instance (delete should never be called directly)

Operation name: OMDefaultInBound

Initializer: itsDefaultProvidedInterface(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMDefaultInBound()
Return Type:
Description: Initialize

Operation name: scheduleTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: scheduleTimeout(OxfTimeUnit /*delay*/,char /*targetStateName*/)
Return Type: IOxfTimeout
Description: schedule a timeout to be consumed by the reactive instance.

Argument information for Operation scheduleTimeout

Name	Type	Direction
/*delay*/	OxfTimeUnit	In
/*targetStateName*/	char	In

Operation name: handleTrigger

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: handleTrigger(IOxfEvent /*ev*/)
Return Type: void
Description: consume a triggered operation (synchronous event)

Argument information for Operation handleTrigger

Name	Type	Direction
/*ev*/	IOxfEvent	In

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: false

Static: false
Virtual: false
Visibility: public
Signature: send(IOxfEvent ev)
Return Type: bool
Description: send the specified event to the instance active context queue

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In

Operation name: getActiveContext

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getActiveContext()
Return Type: IOxfActive
Description: Get the provided interface active context

Operation name: setActiveContext

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setActiveContext(IOxfActive /**/)
Return Type: void
Description: Unused, needed to realize the entire IOxfReactive interface

Argument information for Operation setActiveContext

Name	Type	Direction
/**/	IOxfActive	In

Operation name: getCurrentEvent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getCurrentEvent()
Return Type: IOxfEvent
Description: Get the provided interface current event

Operation name: popNullTransition

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: popNullTransition()
Return Type: void
Description: signal that a null transition was taken (called by the generated code)

Operation name: pushNullTransition

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: pushNullTransition()
Return Type: void
Description: signal that there is a null transition to be taken (called by the generated code)

Operation name: endBehavior

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: endBehavior(char* /*theTerminator*/)
Return Type: void
Description: signal that the reactive instance reached a terminate connector

Argument information for Operation endBehavior

Name	Type	Direction
/*theTerminator*/	char*	In

Operation name: cancelTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cancelTimeout(IOxfTimeout /*timeout*/)
Return Type: bool
Description: cleanup references to the specified timeout
return true if the reference was removed.

Argument information for Operation cancelTimeout

Name	Type	Direction
/*timeout*/	IOxfTimeout	In

Operation name: handleNotConsumed

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: handleNotConsumed(IOxfEvent /*ev*/,EventNotConsumedReason /*reason*/)
Return Type: void
Description: react to an event that was not consumed.
note that the event can be allocated on the stack.

Argument information for Operation handleNotConsumed

Name	Type	Direction
/*ev*/	IOxfEvent	In
/*reason*/	EventNotConsumedReason	In

Generalization information for Class OMDefaultInBound

Generalization name: IOxfReactive

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfReactive	IOxfReactive	OMDefaultInBound

Relation information for Class OMDefaultInBound

Relation name: itsDefaultProvidedInterface

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: itsDefaultProvidedInterface
LinkName:
RoleName: itsDefaultProvidedInterface
Type: Association
Description: The port provided interface

Name	Inverse	Source	Target
itsDefaultProvidedInterface		OMDefaultInBound	IOxfReactive

Class name: OMDefaultOutBound

Description:default out-bound port to a reactive instance.
forward the events to the reactive target.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMDefaultOutBound

Operation name: handleEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: handleEvent(IOxfEvent /*ev*/)
Return Type: TakeEventStatus
Description: consume an event

Argument information for Operation handleEvent

Name	Type	Direction
/*ev*/	IOxfEvent	In

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: send(IOxfEvent ev,IOxfEventGenerationParams params)
Return Type: bool
Description: send an event to the active context queue

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In
params	IOxfEventGenerationParams	In

Operation name: startBehavior

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public

Signature: startBehavior()
Return Type: bool
Description: initialize the reactive instance state machine

Operation name: destroy

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: destroy()
Return Type: void
Description: destroy the reactive instance (delete should never be called directly)

Operation name: OMDefaultOutBound

Initializer: itsDefaultRequiredInterface(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMDefaultOutBound()
Return Type:
Description: Initialize

Operation name: handleTrigger

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: handleTrigger(IOxfEvent /*ev*/)
Return Type: void
Description: consume a triggered operation (synchronous event)

Argument information for Operation handleTrigger

Name	Type	Direction
/*ev*/	IOxfEvent	In

Operation name: scheduleTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: scheduleTimeout(OxfTimeUnit /*delay*/,char /*targetStateName*/)

Return Type: IOxfTimeout
Description: schedule a timeout to be consumed by the reactive instance.

Argument information for Operation scheduleTimeout

Name	Type	Direction
/*delay*/	OxfTimeUnit	In
/*targetStateName*/	char	In

Operation name: send

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: send(IOxfEvent ev)
Return Type: bool
Description: send the specified event to the instance active context queue

Argument information for Operation send

Name	Type	Direction
ev	IOxfEvent	In

Operation name: getActiveContext

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getActiveContext()
Return Type: IOxfActive
Description: Unused, needed to realize the entire IOxfReactive interface

Operation name: getCurrentEvent

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getCurrentEvent()
Return Type: IOxfEvent
Description: Unused, needed to realize the entire IOxfReactive interface

Operation name: setActiveContext

Initializer:
Const: false
Trigger: false

Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setActiveContext(IOxfActive /**/)
Return Type: void
Description: Unused, needed to realize the entire IOxfReactive interface

Argument information for Operation setActiveContext

Name	Type	Direction
/**/	IOxfActive	In

Operation name: popNullTransition

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: popNullTransition()
Return Type: void
Description: signal that a null transition was taken (called by the generated code)

Operation name: pushNullTransition

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: pushNullTransition()
Return Type: void
Description: signal that there is a null transition to be taken (called by the generated code)

Operation name: endBehavior

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: endBehavior(char* /*theTerminator*/)
Return Type: void
Description: signal that the reactive instance reached a terminate connector

Argument information for Operation endBehavior

Name	Type	Direction
/*theTerminator*/	char*	In

Operation name: cancelTimeout

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cancelTimeout(IOxfTimeout /*timeout*/)
Return Type: bool
Description: cleanup references to the specified timeout
return true if the reference was removed.

Argument information for Operation cancelTimeout

Name	Type	Direction
/*timeout*/	IOxfTimeout	In

Operation name: handleNotConsumed

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: handleNotConsumed(IOxfEvent /*ev*/,EventNotConsumedReason /*reason*/)
Return Type: void
Description: react to an event that was not consumed.
note that the event can be allocated on the stack.

Argument information for Operation handleNotConsumed

Name	Type	Direction
/*ev*/	IOxfEvent	In
/*reason*/	EventNotConsumedReason	In

Generalization information for Class OMDefaultOutBound

Generalization name: IOxfReactive

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfReactive	IOxfReactive	OMDefaultOutBound

Relation information for Class OMDefaultOutBound

Relation name: itsDefaultRequiredInterface

Symmetric: false
Multiplicity: 1

Qualifier:
Visibility: public
Label: itsDefaultRequiredInterface
LinkName:
RoleName: itsDefaultRequiredInterface
Type: Association
Description: The port required interface

Name	Inverse	Source	Target
itsDefaultRequiredInterface		OMDefaultOutBound	IOxfReactive

Class name: OMDefaultReactivePort

Description:default reactive in-out port
Active: false
Behavior Overridden: false
Composite: true
Reactive: false

Operation information for Class: OMDefaultReactivePort

Operation name: getInBound

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getInBound()
Return Type: OMDefaultInBound
Description: Get the in bound port

Operation name: getOutBound

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getOutBound()
Return Type: OMDefaultOutBound
Description: Get the out bound port

Operation name: getItsDefaultProvidedInterface

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getItsDefaultProvidedInterface()

Return Type: IOxfReactive
Description: Get the provided interface

Operation name: OMDefaultReactivePort

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMDefaultReactivePort()
Return Type:
Description: Initialize

Operation name: getItsDefaultRequiredInterface

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getItsDefaultRequiredInterface()
Return Type: IOxfReactive
Description: Get the required interface

Operation name: setItsDefaultProvidedInterface

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setItsDefaultProvidedInterface(IOxfReactive reactive)
Return Type: void
Description: Set the provided interface

Argument information for Operation setItsDefaultProvidedInterface

Name	Type	Direction
reactive	IOxfReactive	In

Operation name: setItsDefaultRequiredInterface

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setItsDefaultRequiredInterface(IOxfReactive reactive)
Return Type: void

Description: Set the Required interface

Argument information for Operation setItsDefaultRequiredInterface

Name	Type	Direction
reactive	IOxfReactive	In

Relation information for Class OMDefaultReactivePort

Relation name: InBound

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: InBound
LinkName:
RoleName: InBound
Type: Composition
Description: in port

Relation name: OutBound

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: OutBound
LinkName:
RoleName: OutBound
Type: Composition
Description: out port

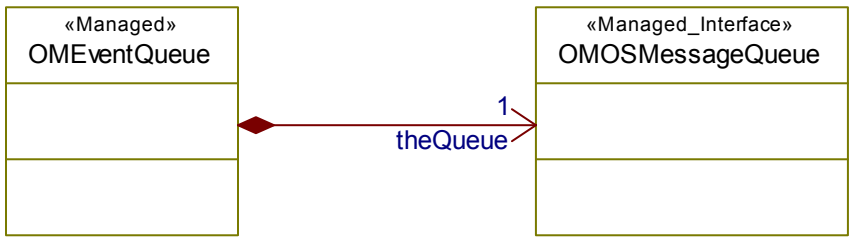
Name	Inverse	Source	Target
InBound		OMDefaultReactivePort	OMDefaultInBound
OutBound		OMDefaultReactivePort	OMDefaultOutBound

Package: RTOSWrappers

Object Model Diagram Information

Object Model Diagram name: The event queue

Description: The event queue architecture



Class Information for Package: RTOSWrappers

Class name: OMEventQueue

Description:An instantiation class of the event queue wrapper over the RTOS message queue.
The message type is IOxfEvent*.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMEventQueue

Operation name: OMEventQueue

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMEventQueue(long messageQueueSize,bool dynamicMessageQueue)
Return Type:
Description: Initialize

Argument information for Operation OMEventQueue

Name	Type	Direction
messageQueueSize	long	In
dynamicMessageQueue	bool	In

Operation name: OMEventQueue

Initializer: theQueue(0)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMEventQueue()
Return Type:
Description: Initialize without creating of the RTOS queue

Operation name: ~OMEventQueue

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMEventQueue()
Return Type:
Description: Cleanup

Operation name: cleanup

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cleanup()
Return Type: void
Description: cleanup the RTOS resources

Operation name: get

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: get()
Return Type: IOxfEvent
Description: Get an event from the queue

Operation name: getMessageList

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getMessageList(OMList<IOxfEvent*> & l)
Return Type: void
Description: Get a list of the events in the queue

Argument information for Operation getMessageList

Name	Type	Direction
l	OMList<IOxfEvent*> & %s	Out

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: Get the RTOS queue

Operation name: init

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: init(long messageQueueSize,bool dynamicMessageQueue)
Return Type: void
Description: Initialize the RTOS queue

Argument information for Operation init

Name	Type	Direction
messageQueueSize	long	In
dynamicMessageQueue	bool	In

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: isEmpty()
Return Type: bool
Description: Check if the queue is empty

Operation name: isFull

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: isFull()
Return Type: bool
Description: Check if the queue is full (dynamic queues are never full)

Operation name: pend

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: pend()
Return Type: void
Description: Block on the queue until a message arrives (non-blocking if there are messages in the queue)

Operation name: put

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: put(IOxfEvent* ev)
Return Type: bool
Description: Put an event into the queue

Argument information for Operation put

Name	Type	Direction
ev	IOxfEvent*	In

Operation name: putMessage

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: putMessage(IOxfEvent* ev,IOxfEventGenerationParams params)
Return Type: bool
Description: Put a event into the queue

Argument information for Operation putMessage

Name	Type	Direction
ev	IOxfEvent*	In
params	IOxfEventGenerationParams	In

Relation information for Class OMEventQueue

Relation name: theQueue

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theQueue
LinkName:
RoleName: theQueue
Type: Composition
Description: The RTOS message queue

Name	Inverse	Source	Target
theQueue		OMEventQueue	OMOSMessageQueue

Class name: OMTMMessageQueue

Description:A typed (Msg*) RTOS message queue (single message type)

Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMTMMessageQueue

Operation name: OMTMMessageQueue

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMTMMessageQueue(long messageQueueSize,bool dynamicMessageQueue)
Return Type:
Description: Initialize the queue

Argument information for Operation OMTMMessageQueue

Name	Type	Direction
messageQueueSize	long	In
dynamicMessageQueue	bool	In

Operation name: get

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: get()
Return Type: Msg*
Description: Get a message from the queue

Operation name: getMessageList

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getMessageList(OMList<Msg*> & l)
Return Type: void
Description: Get a list of the messages in the queue

Argument information for Operation getMessageList

Name	Type	Direction
l	OMList<Msg*> & %s	Out

Operation name: getOsHandle

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getOsHandle()
Return Type: void *
Description: Get the RTOS queue

Operation name: isEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: isEmpty()
Return Type: bool
Description: Check if the queue is empty

Operation name: isFull

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: isFull()
Return Type: bool
Description: Check if the queue is full (dynamic queues are never full)

Operation name: pend

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: pend()
Return Type: void
Description: Block on the queue until a message arrives (non-blocking if there are messages in the queue)

Operation name: putMessage

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: true
Visibility: public
Signature: putMessage(Msg* m,IOxfEventGenerationParams params)
Return Type: bool
Description: Put a message into the queue

Argument information for Operation putMessage

Name	Type	Direction
m	Msg*	In
params	IOxfEventGenerationParams	In

Operation name: ~OMTMMessageQueue

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMTMMessageQueue()
Return Type:
Description: Cleanup

Operation name: put

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: put(Msg* m)
Return Type: bool
Description: Put a message into the queue

Argument information for Operation put

Name	Type	Direction
m	Msg*	In

Operation name: OMTMMessageQueue

Initializer: theQueue(0)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMTMMessageQueue()
Return Type:
Description: Initialize without creating of the RTOS queue

Operation name: init

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: init(long messageQueueSize,bool dynamicMessageQueue)
Return Type: void
Description: Initialize the RTOS queue

Argument information for Operation init

Name	Type	Direction
messageQueueSize	long	In
dynamicMessageQueue	bool	In

Operation name: cleanup

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cleanup()
Return Type: void
Description: cleanup the RTOS resources

Relation information for Class OMTMMessageQueue

Relation name: theQueue

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: theQueue
LinkName:
RoleName: theQueue
Type: Composition
Description: The RTOS message queue

Name	Inverse	Source	Target
theQueue		OMTMMessageQueue	OMOSMessageQueue

Class name: OMOSEventGenerationParams

Description:Default implementation of IOxfEventGenerationParams.
Can be replaced by creating a new implementation with the same name and add
OM_ADAPTER_OS_EVEMT_GENERATION_PARAMS to the compilation switches.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMOSEventGenerationParams

Attribute Name: fromISR

Default Value: false
Static: false
Visibility: public
Type: bool
Stereotype:
Description: Should be set to true when the event is generated by an ISR

Attribute Name: sender

Default Value: 0
Static: false
Visibility: public
Type: void *
Stereotype:
Description: The sender of the event (instrumentation support)

Operation information for Class: OMOSEventGenerationParams

Operation name: OMOSEventGenerationParams

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMOSEventGenerationParams()
Return Type:
Description: Default initialization

Operation name: OMOSEventGenerationParams

Initializer: fromISR(genFromISR), sender(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMOSEventGenerationParams(bool genFromISR)
Return Type:
Description: Initialize based on the genFromISR attribute

Argument information for Operation OMOSEventGenerationParams

Name	Type	Direction
genFromISR	bool	In

Operation name: OMOSEventGenerationParams

Initializer: fromISR(false), sender(theSender)
Const: false
Trigger: false

Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMOSEventGenerationParams(void * theSender)
Return Type:
Description: Initialize based on the sender attribute

Argument information for Operation OMOSEventGenerationParams

Name	Type	Direction
theSender	void *	In

Operation name: ~OMOSEventGenerationParams

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMOSEventGenerationParams()
Return Type:
Description: Cleanup

Generalization information for Class OMOSEventGenerationParams

Generalization name: IOxfEventGenerationParams

Description:
Virtual: false
Visibility: public
Extension Point:

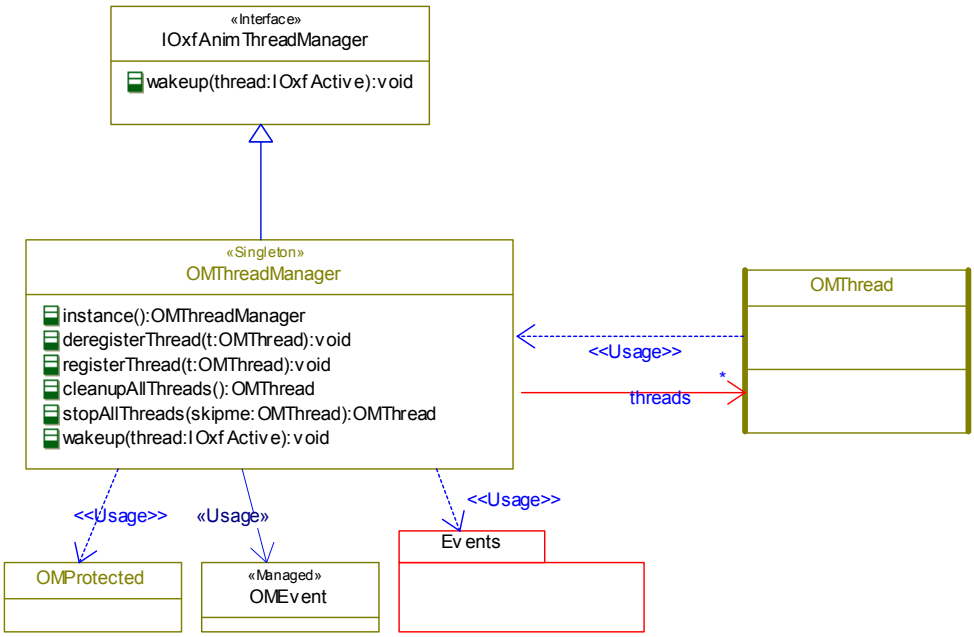
Name	Base	Derived
IOxfEventGenerationParams	IOxfEventGenerationParams	OMOSEventGenerationParams

Package: ResourceManagement

Object Model Diagram Information

Object Model Diagram name: thread manager overview

Description: The thread manager design overview



Class Information for Package: ResourceManagement

Class name: OMHandleCloser

Description:Provide mechanism for cleanup after thread deletion
Active: false
Behavior Overridden: false
Composite: true
Reactive: true

Attribute Information for Class: OMHandleCloser

Attribute Name: doCloseHandle

Default Value: doCloseHandlePtr
Static: false
Visibility: private
Type: closeHandleFunc
Stereotype:
Description: The cleanup callback function

Attribute Name: _singletonDestroyed

Default Value: false
Static: true
Visibility: private
Type: bool
Stereotype:
Description: Indicate that the singleton was destroyed (while exit())

Operation information for Class: OMHandleCloser

Operation name: OMHandleCloser

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMHandleCloser(closeHandleFunc doCloseHandlePtr)
Return Type:
Description: Initialize

Argument information for Operation OMHandleCloser

Name	Type	Direction
doCloseHandlePtr	closeHandleFunc	In

Operation name: ~OMHandleCloser

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMHandleCloser()
Return Type:
Description: Cleanup

Operation name: genCloseEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: genCloseEvent(void * hObject)
Return Type: void
Description: sends event to the HandleCloser thread to delete hObject thread

Argument information for Operation genCloseEvent

Name	Type	Direction
hObject	void *	In

Operation name: getInstance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: private
Signature: getInstance(closeHandleFunc doCloseHandlePtr)
Return Type: OMHandleCloser &

Description: Initialize and get the singleton

Argument information for Operation getInstance

Name	Type	Direction
doCloseHandlePtr	closeHandleFunc	In

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance()
Return Type: OMHandleCloser
Description: Get the singleton

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance(closeHandleFunc doCloseHandlePtr)
Return Type: OMHandleCloser
Description: returns handle closer instance and initializes its doCloseHandle function ptr

Argument information for Operation instance

Name	Type	Direction
doCloseHandlePtr	closeHandleFunc	In

Operation name: sendCloseHandleCloserEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: sendCloseHandleCloserEvent()
Return Type: void
Description: Send the OMCloseHandleEvent to itself

Operation name: startBehavior

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: true
Visibility: public
Signature: startBehavior()
Return Type: bool
Description: Start the statechart

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: operator=(OMHandleCloser other)
Return Type: OMHandleCloser
Description: disable assignment operator

Argument information for Operation operator=

Name	Type	Direction
other	OMHandleCloser	In

Operation name: OMHandleCloser

Initializer: OMReactive(), doCloseHandle(0)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: OMHandleCloser(OMHandleCloser other)
Return Type:
Description: disable copy constructor

Argument information for Operation OMHandleCloser

Name	Type	Direction
other	OMHandleCloser	In

EventReception information for Class OMHandleCloser

Event Reception name: OMCloseHandleEvent

Signature: OMCloseHandleEvent(void * handle)
Description: The handled event

Argument information for Event OMCloseHandleEvent

Name	Type	Direction
handle	void *	In

Type information for Class OMHandleCloser

Type name: closeHandleFunc

Description: The RTOS callback function that perform the RTOS thread cleanup
Kind: Language
Declaration: typedef void (* %s)(void *)

Generalization information for Class OMHandleCloser

Generalization name: OMReactive

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMReactive	OMReactive	OMHandleCloser

Relation information for Class OMHandleCloser

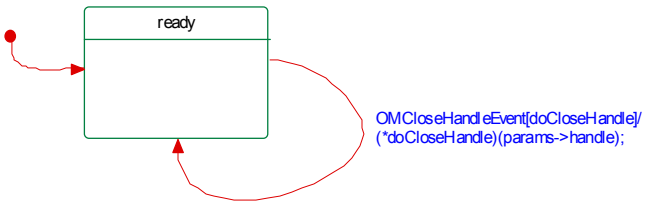
Relation name: thread

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: thread
LinkName:
RoleName: thread
Type: Composition
Description: The handle closer context

Name	Inverse	Source	Target
thread		OMHandleCloser	OMThread

Statechart information for Class OMHandleCloser

Description:
Overridden: false



State information

State: ROOT

Default Transition information for State ROOT

Description:

Overridden: true
Label:

Incoming Transition information for State ROOT

Description:
Overridden: true
Label:

Outgoing Transition information for State ROOT

Description:
Overridden: true
Label:

State information

State: ready

Class name: OMThreadManager

Description:This class is responsible for managing the living threads list and for cleanup of these threads on close of the application.
It is used by RTOS that do not have automatic process cleanup such as VxWorks.
The cleanup is also used to cleanup the framework threads on unload of the framework DLL.
Compile the oxf with the OM_NO_APPLICATION_TERMINATION_SUPPORT compilation flag to disable the management of the list.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMThreadManager

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: instance()
Return Type: OMThreadManager
Description: Get the singleton instance

Operation name: deregisterThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: deregisterThread(OMThread t)
Return Type: void

Description: Deregister an about to die thread

Argument information for Operation deregisterThread

Name	Type	Direction
t	OMThread	In

Operation name: registerThread

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: registerThread(OMThread t)
Return Type: void
Description: register a new thread

Argument information for Operation registerThread

Name	Type	Direction
t	OMThread	In

Operation name: cleanupAllThreads

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: cleanupAllThreads()
Return Type: OMThread
Description: Cleanup all the live threads on close of the application.
Cleanup all the registered threads except for:
- OMMainThread::instance()
- threads that return false in allowDeleteInThreadsCleanup() call
- the thread context that this call is invoked on

Should be called only in OSAL cleanup when the RTOS exit() doesn't perform the cleanup
return the thread context

Operation name: stopAllThreads

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: stopAllThreads(OMThread skipme)
Return Type: OMThread
Description: Stop all the live threads before unload of the framework as a DLL.

stop all the registered threads
except the skipme thread and the context thread
return the context thread

Argument information for Operation stopAllThreads

Name	Type	Direction
skipme	OMThread	In

Operation name: OMThreadManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: protected
Signature: OMThreadManager()
Return Type:
Description: Initialization

Operation name: wakeup

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: wakeup(IOxfActive thread)
Return Type: void
Description: Wakeup the specified thread (the thread is supposed to be waiting for events)

Argument information for Operation wakeup

Name	Type	Direction
thread	IOxfActive	In

Type information for Class OMThreadManager

Type name: Guard

Description: The manager guard declaration
Kind: Language
Declaration: OMDECLARE_GUARDED // %s

Generalization information for Class OMThreadManager

Generalization name: IOxfAnimThreadManager

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfAnimThreadManager	IOxfAnimThreadManager	OMThreadManager

Relation information for Class OMThreadManager

Relation name: threads

Symmetric: false
Multiplicity: *
Qualifier:
Visibility: public
Label: threads
LinkName:
RoleName: threads
Type: Association
Description: The list of live threads in the system

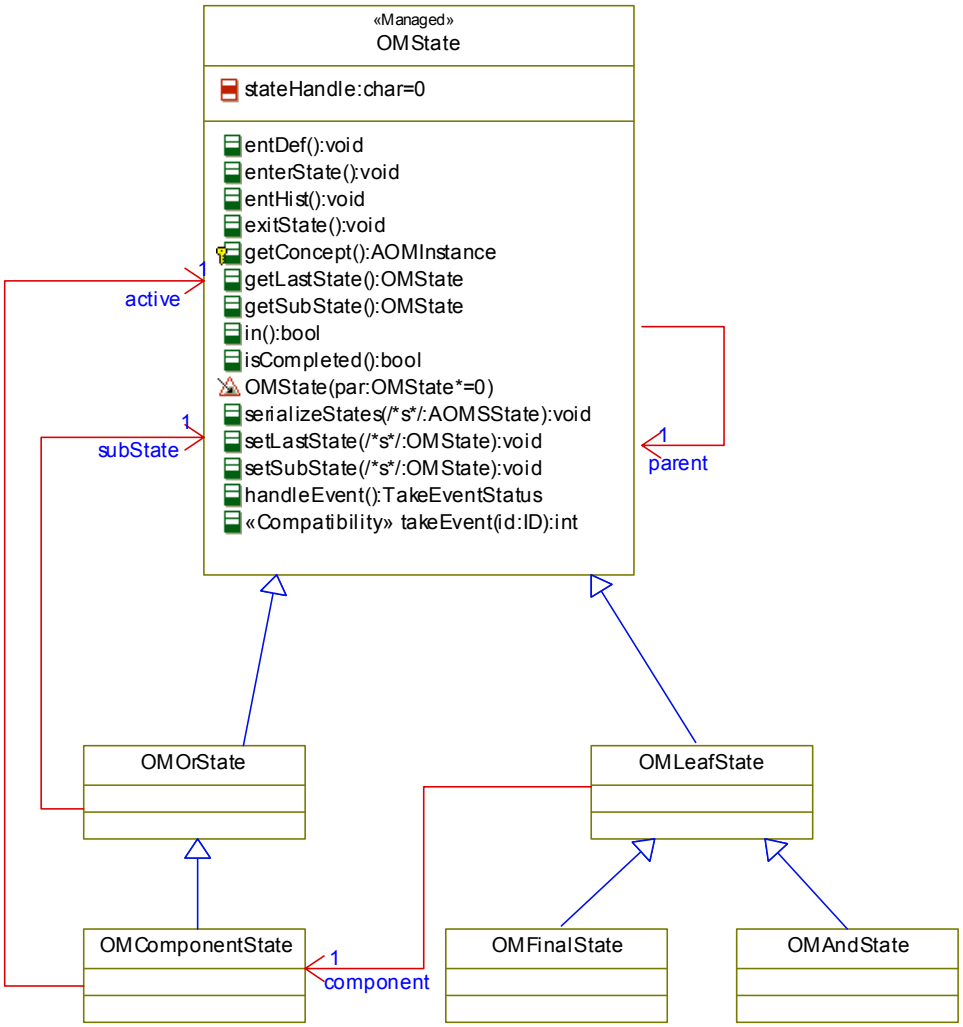
Name	Inverse	Source	Target
threads		OMThreadManager	OMThread

Package: ReusableBaseStates

Object Model Diagram Information

Object Model Diagram name: base states

Description: The states hierarchy.



Class Information for Package: ReusableBaseStates

Class name: OMState

Description:The base state class
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMState

Attribute Name: stateHandle

Default Value: 0
Static: false
Visibility: public
Type: char
Stereotype:
Description: The state name (for instrumentation)

Operation information for Class: OMState

Operation name: entDef

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: entDef()
Return Type: void
Description: Enter this state via the default transition

Operation name: enterState

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: enterState()
Return Type: void
Description: Enter this state

Operation name: entHist

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: entHist()
Return Type: void
Description: Enter a state via a history connector.

Operation name: exitState

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: exitState()
Return Type: void
Description: Exit from this state

Operation name: getConcept

Initializer:
Const: true
Trigger: false

Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: getConcept()
Return Type: AOMInstance
Description: Get the reactive owner of this state

Operation name: getLastState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getLastState()
Return Type: OMState
Description: Get the last active state in this sub graph (for history connector support)

Operation name: getSubState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: getSubState()
Return Type: OMState
Description: Get the active sub state

Operation name: in

Initializer:
Const: false
Trigger: false
Abstract: true
Static: false
Virtual: false
Visibility: public
Signature: in()
Return Type: bool
Description: Check is this state is in the active states graph

Operation name: isCompleted

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: isCompleted()
Return Type: bool
Description: Check is this state is completed (the sub graph reached a final state)

Operation name: OMState

Initializer: parent(par)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMState(OMState* par)
Return Type:
Description: Initialize

Argument information for Operation OMState

Name	Type	Direction
par	OMState*	In

Operation name: serializeStates

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: serializeStates(AOMSState /*s*/)
Return Type: void
Description: Serialize the state graph (instrumentation)

Argument information for Operation serializeStates

Name	Type	Direction
/*s*/	AOMSState	InOut

Operation name: setLastState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: setLastState(OMState /*s*/)
Return Type: void
Description: Set the last active state in this sub graph (for history connector support)

Argument information for Operation setLastState

Name	Type	Direction
/*s*/	OMState	In

Operation name: setSubState

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: setSubState(OMState /*s*/)
Return Type: void
Description: Set the active sub state

Argument information for Operation setSubState

Name	Type	Direction
/*s*/	OMState	In

Operation name: handleEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: handleEvent()
Return Type: TakeEventStatus
Description: Handle the current event (of the reactive owner)

Operation name: takeEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: takeEvent(ID id)
Return Type: int
Description: 5.X compatibility API: handle the current event

Argument information for Operation takeEvent

Name	Type	Direction
id	ID	In

Relation information for Class OMState

Relation name: parent

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: parent
LinkName:
RoleName: parent

Type: Association
Description: The parent state

Name	Inverse	Source	Target
parent		OMState	OMState

Class name: OMOrState

Description:A non-concurrent composite state
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMOrState

Operation name: entDef

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: entDef()
Return Type: void
Description: Enter this state via the default transition

Operation name: enterState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: enterState()
Return Type: void
Description: Enter this state

Operation name: exitState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: exitState()
Return Type: void
Description: Exit from this state

Operation name: in

Initializer:
Const: false

Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: in()
Return Type: bool
Description: Check is this state is in the active states graph

Operation name: OMOrState

Initializer: OMState(par), subState(NULL)
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMOrState(OMState par)
Return Type:
Description: Initialize

Argument information for Operation OMOrState

Name	Type	Direction
par	OMState	In

Operation name: serializeStates

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: serializeStates(AOMSState aomArg(s))
Return Type: void
Description: Serialize the state graph (instrumentation)

Argument information for Operation serializeStates

Name	Type	Direction
aomArg(s)	AOMSState	InOut

Generalization information for Class OMOrState

Generalization name: OMState

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMState	OMState	OMOrState

Relation information for Class OMOrState

Relation name: subState

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: subState
LinkName:
RoleName: subState
Type: Association
Description: The direct sub state that is part of the active graph

Name	Inverse	Source	Target
subState		OMOrState	OMState

Class name: OMComponentState

Description:A part of an AND state or the root state
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMComponentState

Operation name: enterState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: enterState()
Return Type: void
Description: Enter into this state

Operation name: in

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: in()
Return Type: bool
Description: Check is this state is in the active states graph

Operation name: OMComponentState

Initializer: OMOrState(par), active(NULL)
Const: false
Trigger: false
Abstract: false

Static: false
Virtual: false
Visibility: public
Signature: OMComponentState(OMState par)
Return Type:
Description: Initialize

Argument information for Operation OMComponentState

Name	Type	Direction
par	OMState	In

Operation name: handleEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: handleEvent()
Return Type: TakeEventStatus
Description: Handle the current event

Operation name: takeEvent

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: takeEvent(ID /*id*/)
Return Type: int
Description: 5.X compatibility API: handle the current event

Argument information for Operation takeEvent

Name	Type	Direction
/*id*/	ID	In

Generalization information for Class OMComponentState

Generalization name: OMOrState

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMOrState	OMOrState	OMComponentState

Relation information for Class OMComponentState

Relation name: active

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: active
LinkName:
RoleName: active
Type: Association
Description: The active sub state

Name	Inverse	Source	Target
active		OMComponentState	OMState

Class name: OMLeafState

Description:A simple state that doesn't contain additional states
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMLeafState

Operation name: entDef

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: entDef()
Return Type: void
Description: Take the state default transition

Operation name: enterState

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: enterState()
Return Type: void
Description: Enter the state

Operation name: exitState

Initializer:
Const: false
Trigger: false
Abstract: false

Static: false
Virtual: true
Visibility: public
Signature: exitState()
Return Type: void
Description: Exit the state

Operation name: in

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: in()
Return Type: bool
Description: Check is this state is in the active states graph

Operation name: OMLeafState

Initializer: OMState(par), component(static_cast<OMComponentState*>(cmp))

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMLeafState(OMState par,OMState cmp)
Return Type:
Description: Initialize

Argument information for Operation OMLeafState

Name	Type	Direction
par	OMState	In
cmp	OMState	In

Operation name: serializeStates

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: serializeStates(AOMSState aomArg(s))
Return Type: void
Description: Serialize the state graph (instrumentation)

Argument information for Operation serializeStates

Name	Type	Direction
aomArg(s)	AOMSState	InOut

Generalization information for Class OMLeafState

Generalization name: OMState

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMState	OMState	OMLeafState

Relation information for Class OMLeafState

Relation name: component

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: component
LinkName:
RoleName: component
Type: Association
Description:

Name	Inverse	Source	Target
component		OMLeafState	OMComponentState

Class name: OMAndState

Description:A concurrent state
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMAndState

Attribute Name: _lock

Default Value:
Static: false
Visibility: private
Type: bool
Stereotype:
Description: A locked flag

Operation information for Class: OMAndState

Operation name: OMAndState

Initializer: OMLeafState(par, cmp), _lock(false)

Const: false
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: public
Signature: OMAndState(OMState par,OMState cmp)
Return Type:
Description: Initialize

Argument information for Operation OMAndState

Name	Type	Direction
par	OMState	In
cmp	OMState	In

Operation name: lock

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: lock()
Return Type: void
Description: Set the lock

Operation name: unlock

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: unlock()
Return Type: void
Description: Unset the lock

Generalization information for Class OMAndState

Generalization name: OMLeafState

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMLeafState	OMLeafState	OMAndState

Class name: OMFinalState

Description:UML Final state
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMFinalState

Operation name: OMFinalState

Initializer: OMLeafState(par, cmp) , concept(cpt)

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMFinalState(IOxfReactive cpt,OMState par,OMState cmp,char* aomArg(hdl))
Return Type:
Description: Initialize

Argument information for Operation OMFinalState

Name	Type	Direction
cpt	IOxfReactive	In
par	OMState	In
cmp	OMState	In
aomArg(hdl)	char*	In

Operation name: getConcept

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: getConcept()
Return Type: AOMInstance
Description: Get the reactive owner

Generalization information for Class OMFinalState

Generalization name: OMLeafState

Description:
Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
OMLeafState	OMLeafState	OMFinalState

Relation information for Class OMFinalState

Relation name: concept

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public

Label: concept
LinkName:
RoleName: concept
Type: Association
Description: The reactive owner

Name	Inverse	Source	Target
concept		OMFinalState	IOxfReactive

Type information for Package ReusableBaseStates

Type name: eventConsumed

Description: Make IReactive::TakeEventStatus literals available in state classes
Kind: Language
Declaration: #define %s IOxfReactive::%s

Type name: eventNotConsumed

Description: Make IReactive::TakeEventStatus literals available in state classes
Kind: Language
Declaration: #define %s IOxfReactive::%s

Type name: instanceReachTerminate

Description: Make IReactive::TakeEventStatus literals available in state classes
Kind: Language
Declaration: #define %s IOxfReactive::%s

Type name: instanceUnderDestruction

Description: Make IReactive::TakeEventStatus literals available in state classes
Kind: Language
Declaration: #define %s IOxfReactive::%s

Package: SelectiveInclude

Type information for Package SelectiveInclude

Type name: ANIM_INCLUDE

Description: Compilation dependent include settings for animation support
Kind: Language
Declaration: #ifdef ANIM_USE_IOSTREAM
#ifndef OM_FROCE_IOSTREAM
#define OM_FROCE_IOSTREAM
#endif // !OM_FROCE_IOSTREAM
#elif (defined ANIM_USE_STDIO)
#ifdef OM_FROCE_IOSTREAM
#undef OM_FROCE_IOSTREAM
#endif // OM_FROCE_IOSTREAM
#ifndef OM_FROCE_STDIO
#define OM_FROCE_STDIO
#endif // !OM_FROCE_STDIO
#endif // ANIM_USE_IOSTREAM

Type name: OM_FROCE_IOSTREAM

Description: Forcing usage of iostream
Has higher priority than OM_FORCE_STDIO
Kind: Language
Declaration: #ifdef OM_FROCE_IOSTREAM
// force iostream settings by disable of stdio settings
// and defining USE_IOSTREAM
#ifdef OM_FROCE_STDIO
#undef OM_FROCE_STDIO
#endif // OM_FROCE_STDIO

#ifdef OM_FORCE_STDIO
#undef OM_FORCE_STDIO
#endif // OM_FORCE_STDIO

#ifndef USE_IOSTREAM
#define USE_IOSTREAM
#endif // !USE_IOSTREAM
#endif //OM_FROCE_IOSTREAM

Type name: OM_FORCE_STDIO

Description: Forcing usage of stdio
Kind: Language
Declaration: #if (defined OM_FROCE_STDIO || defined %s)
// OM_FROCE_STDIO is kept for backward compatibility
#ifdef USE_IOSTREAM
#undef USE_IOSTREAM
#endif // USE_IOSTREAM
#ifndef USE_STDIO
#define USE_STDIO
#endif // USE_STDIO
#endif // if (defined OM_FROCE_STDIO || defined %s)

Type name: favor_iostream

Description: if BOTH USE_IOSTREAM and USE_STDIO are defined
use iostreams
Kind: Language
Declaration: #if (defined(USE_IOSTREAM) && defined(USE_STDIO))
#undef USE_STDIO
#endif // (defined(USE_IOSTREAM) && defined(USE_STDIO))

Type name: IOSTREAM_INCLUDE

Description: Compilation dependent include to iostream
Kind: Language
Declaration: #ifdef USE_IOSTREAM
#if (defined(OM_STL) && !defined(OM_NO_STD_STRING))
#include <iostream>
#else
#include <iostream.h>
#endif // (defined(OM_STL) && !defined(OM_NO_STD_STRING))
#endif // USE_IOSTREAM

Type name: STDLIB_INCLUDE

Description: Compilation dependent include to cstdlib

Kind: Language
Declaration: #ifndef NO_STDLIB
#if (defined(OM_STL) && !defined(OM_NO_STD_STRING))
#include <cstdlib>
#else
#include <stdlib.h>
#endif // (defined(OM_STL) && !defined(OM_NO_STD_STRING))
#endif // NO_STDLIB

Type name: STDIO_INCLUDE

Description: Compilation dependent include to cstdio
Kind: Language
Declaration: #ifndef OM_NO_OS_STDIO_INCLUDE
#ifdef OM_STL
#include <cstdio>
#else
#include <stdio.h>
#endif // OM_STL
#endif // OM_NO_OS_STDIO_INCLUDE

Type name: CTYPE_INCLUDE

Description: Compilation dependent include to cctype
Kind: Language
Declaration: #ifdef OM_STL
#include <cctype>
#else
#include <ctype.h>
#endif // OM_STL

Type name: STRING_INCLUDE

Description: Compilation dependent include to cstring
Kind: Language
Declaration: #if (defined(OM_STL) && !defined(OM_NO_STD_STRING))
#include <cstring>
#else
#include <string.h>
#endif // (defined(OM_STL) && !defined(OM_NO_STD_STRING))

Package: String

Class Information for Package: String

Class name: OMString

Description:A string class, supports the same API as CString.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMString

Attribute Name: count

Default Value: 0

Static: false
Visibility: public
Type: int
Stereotype:
Description: How many chars we currently have (without the '\0')

Attribute Name: defaultBlock

Default Value: 256
Static: true
Visibility: public
Type: int
Stereotype:
Description: the string default size
need to be declared before used (in this file) to avoid compilation issues in some compilers

Attribute Name: size

Default Value: 0
Static: false
Visibility: private
Type: int
Stereotype:
Description: The current allocated memory

Attribute Name: str

Default Value: 0
Static: false
Visibility: private
Type: char*
Stereotype:
Description: Pointer to actual string

Operation information for Class: OMString

Operation name: ~OMString

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMString()
Return Type:
Description: Cleanup

Operation name: CompareNoCase

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: CompareNoCase(char* s2)

Return Type: int
Description: No case compare

Argument information for Operation CompareNoCase

Name	Type	Direction
s2	char*	In

Operation name: CompareNoCase

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: CompareNoCase(OMString s)
Return Type: int
Description: No case compare

Argument information for Operation CompareNoCase

Name	Type	Direction
s	OMString	In

Operation name: Empty

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: Empty()
Return Type: void
Description: empty the string

Operation name: GetBuffer

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: GetBuffer(int /**/)
Return Type: char*
Description: get the string buffer

Argument information for Operation GetBuffer

Name	Type	Direction
/**/	int	In

Operation name: IsEmpty

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: IsEmpty()
Return Type: bool
Description: check if string is empty

Operation name: OMString

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMString(OMString s)
Return Type:
Description: Initialize a string based on another string

Argument information for Operation OMString

Name	Type	Direction
s	OMString	In

Operation name: OMString

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMString(char c)
Return Type:
Description: Initialize a string based on a single character

Argument information for Operation OMString

Name	Type	Direction
c	char	In

Operation name: OMString

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public

Signature: OMString(char* s)
Return Type:
Description: Initialize a string based on another string (C style)

Argument information for Operation OMString

Name	Type	Direction
s	char*	In

Operation name: OMString

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMString()
Return Type:
Description: Initialize an empty string

Operation name: operator const char*

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator const char*()
Return Type:
Description: cast operator

Operation name: operator!=

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator!=(char* c2)
Return Type: bool
Description: not equal test with a C string

Argument information for Operation operator!=

Name	Type	Direction
c2	char*	In

Operation name: operator!=

Initializer:
Const: true
Trigger: false

Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator!=(OMString s2)
Return Type: bool
Description: not equal test with a string

Argument information for Operation operator!=

Name	Type	Direction
s2	OMString	In

Operation name: operator[]

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator[](int i)
Return Type: char
Description: return the character at the given position

Argument information for Operation operator[]

Name	Type	Direction
i	int	In

Operation name: operator+=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator+=(char* s)
Return Type: OMString
Description: Add a C style string to the end of this string

Argument information for Operation operator+=

Name	Type	Direction
s	char*	In

Operation name: operator+=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false

Visibility: public
Signature: operator+=(char c)
Return Type: OMString
Description: Add a character to the end of this string

Argument information for Operation operator+=

Name	Type	Direction
c	char	In

Operation name: operator+=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator+=(OMString s)
Return Type: OMString
Description: Add a string to the end of this string

Argument information for Operation operator+=

Name	Type	Direction
s	OMString	In

Operation name: operator<

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator<(char* c2)
Return Type: bool
Description: Less than test

Argument information for Operation operator<

Name	Type	Direction
c2	char*	In

Operation name: operator<

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator<(OMString s2)
Return Type: bool

Description: Less than test

Argument information for Operation operator<

Name	Type	Direction
s2	OMString	In

Operation name: operator<=

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator<=(char* c2)
Return Type: bool
Description: Less than or equal to test

Argument information for Operation operator<=

Name	Type	Direction
c2	char*	In

Operation name: operator<=

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator<=(OMString s2)
Return Type: bool
Description: Less than or equal to test

Argument information for Operation operator<=

Name	Type	Direction
s2	OMString	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(char* s)
Return Type: OMString
Description: Assign the specified string as the value of this string

Argument information for Operation operator=

Name	Type	Direction
s	char*	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(char c)
Return Type: OMString
Description: Assign the specified character as the value of this string

Argument information for Operation operator=

Name	Type	Direction
c	char	In

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator=(OMString s)
Return Type: OMString
Description: Assign the specified string as the value of this string

Argument information for Operation operator=

Name	Type	Direction
s	OMString	In

Operation name: operator==

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator==(char* c2)
Return Type: bool
Description: Compare this string with the specified string

Argument information for Operation operator==

Name	Type	Direction

c2	char*	In

Operation name: operator==

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator==(OMString s2)
Return Type: bool
Description: Compare this string with the specified string

Argument information for Operation operator==

Name	Type	Direction
s2	OMString	In

Operation name: operator>

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator>(char* c2)
Return Type: bool
Description: Greater than test

Argument information for Operation operator>

Name	Type	Direction
c2	char*	In

Operation name: operator>

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator>(OMString s2)
Return Type: bool
Description: Greater than test

Argument information for Operation operator>

Name	Type	Direction
s2	OMString	In

Operation name: operator>=

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator>=(char* c2)
Return Type: bool
Description: Greater than or equal to test

Argument information for Operation operator>=

Name	Type	Direction
c2	char*	In

Operation name: operator>=

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: operator>=(OMString s2)
Return Type: bool
Description: Greater than or equal to test

Argument information for Operation operator>=

Name	Type	Direction
s2	OMString	In

Operation name: resetSize

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: resetSize(int newSize)
Return Type: void
Description: give string a new larger size
and copy contents to it.

Argument information for Operation resetSize

Name	Type	Direction
newSize	int	In

Operation name: SetAt

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: SetAt(int i,char c)
Return Type: void
Description: set a character at a given position

Argument information for Operation SetAt

Name	Type	Direction
i	int	In
c	char	In

Operation name: setSize

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: setSize(int newSize)
Return Type: void
Description: allocate the string buffer

Argument information for Operation setSize

Name	Type	Direction
newSize	int	In

Operation name: CompareNoCase_

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: private
Signature: CompareNoCase_(char* s1,char* s2)
Return Type: int
Description: No case compare

Argument information for Operation CompareNoCase_

Name	Type	Direction
s1	char*	In
s2	char*	In

Operation name: GetBuffer_

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: GetBuffer_(int newBufferSize)
Return Type: char*
Description: get the string buffer, readjusting its size.

Argument information for Operation GetBuffer_

Name	Type	Direction
newBufferSize	int	In

Package Information

Description: String support

Package: StringFunctions

Package Information

Description: Global string functions

Package: OMString

Operation information for Package OMString

Operation name: operator>=

Static: false
Signature: operator>=(char* c1,OMString s2)
Description: Greater than or equal to compare C string to an OMString

Operation name: operator>

Static: false
Signature: operator>(char* c1,OMString s2)
Description: Greater than compare C string to an OMString

Operation name: operator==

Static: false
Signature: operator==(char* c1,OMString s2)
Description: Equal to compare C string to an OMString

Operation name: operator<=

Static: false
Signature: operator<=(char* c1,OMString s2)
Description: Less than or equal to compare C string to an OMString

Operation name: operator<

Static: false
Signature: operator<(char* c1,OMString s2)
Description: Less than compare C string to an OMString

Operation name: operator+

Static: false
Signature: operator+(OMString s1,OMString s2)
Description: Add stings

Operation name: operator+

Static: false
Signature: operator+(OMString s1,char* s2)
Description: Add stings

Operation name: operator+

Static: false
Signature: operator+(char* s1,OMString s2)
Description: Add stings

Operation name: operator!=

Static: false
Signature: operator!=(char* c1,OMString s2)
Description: not equal test

Operation name: OMDestructiveString2X

Static: false
Signature: OMDestructiveString2X(char* c,OMString /* dummy */)
Description: Instrumentation support, convert a char* to an OMString

Operation name: operator+

Static: false
Signature: operator+(OMString str,char c)
Description: Add a sting and a character

Operation name: operator <<

Static: false
Signature: operator <<(omostream oStream,OMString str)
Description: ostream << OMString operator

Operation name: operator >>

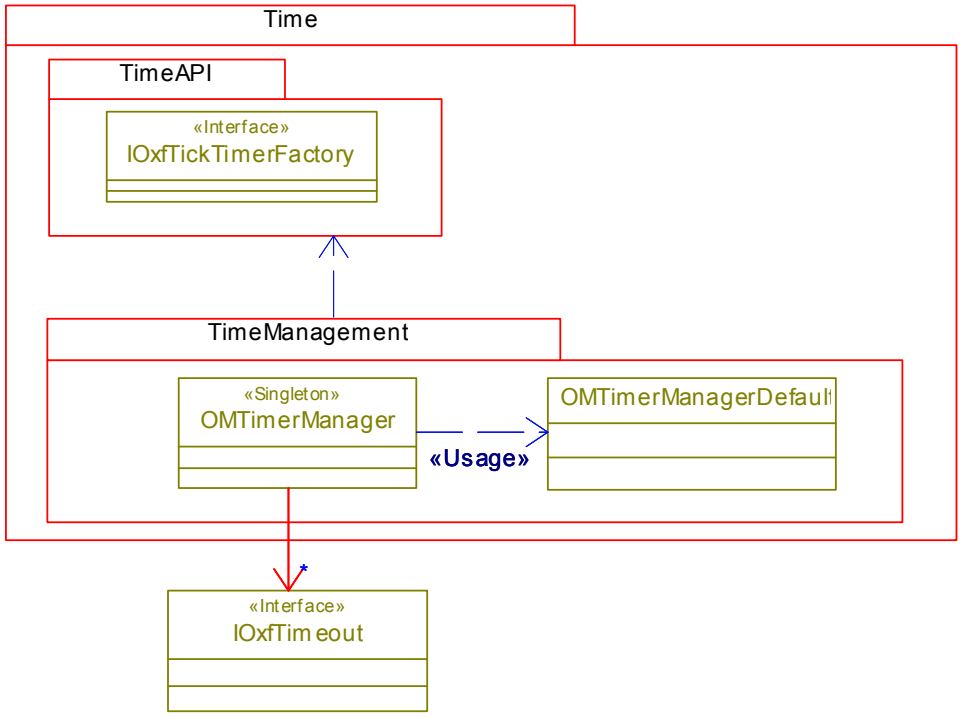
Static: false
Signature: operator >>(omistream iStream,OMString str)
Description: iostream >> OMString operator

Package: Time

Object Model Diagram Information

Object Model Diagram name: Time services overview

Description: Overview on the time related services of the framework



Package Information

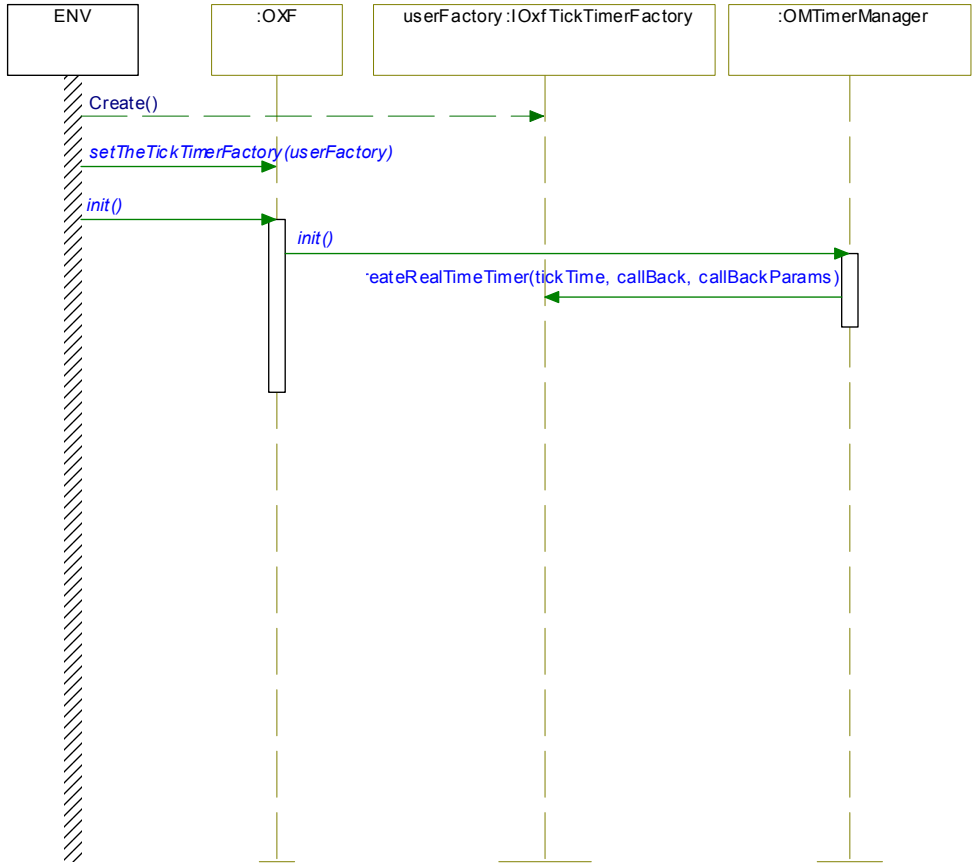
Description: Timing services package

Package: TimeAPI

Sequence Diagram Information

Sequence Diagram name: Usage of the timer factory

Description: This diagram shows how the timer factory can be utilized



Class Information for Package: TimeAPI

Class name: IOxfTickTimerFactory

Description:Low-level timer factory interface.
Enable the user to plug-in its own tick-timers that are created by the implementation of the factory.
The factory enables the user to replace the low-level timer without modifying the adapter timer implementation.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: IOxfTickTimerFactory

Operation name: createRealTimeTimer

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: true
Visibility: public
Signature: createRealTimeTimer(OxfTimeUnit tickTime,TimerManagerCallBack callBack,void * callBackParams)
Return Type: OMOSTimer
Description: create a real-time timer.
the timer should call the TimerManagerCallBack(callBackParams) every tickTime.
The method returns a handle to the timer, so it can be deleted when the timer manager is destroyed.

Argument information for Operation createRealTimeTimer

Name	Type	Direction
tickTime	OxfTimeUnit	In
callBack	TimerManagerCallBack	In
callBackParams	void *	In

Operation name: createSimulatedTimeTimer

Initializer:
Const: true
Trigger: false
Abstract: true
Static: false
Virtual: true
Visibility: public
Signature: createSimulatedTimeTimer(TimerManagerCallBack callBack,void * callBackParams)
Return Type: OMOSTimer
Description: create a simulated-time timer.
the timer should call the TimerManagerCallBack(callBackParams) when the rest of the application is idle.
The method returns a handle to the timer, so it can be deleted when the timer manager is destroyed.

Argument information for Operation createSimulatedTimeTimer

Name	Type	Direction
callBack	TimerManagerCallBack	In
callBackParams	void *	In

Operation name: ~IOxfTickTimerFactory

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~IOxfTickTimerFactory()
Return Type:

Description: Virtual destructor to enable polymorphic deletion

Type information for Class IOxfTickTimerFactory

Type name: TimerManagerCallBack

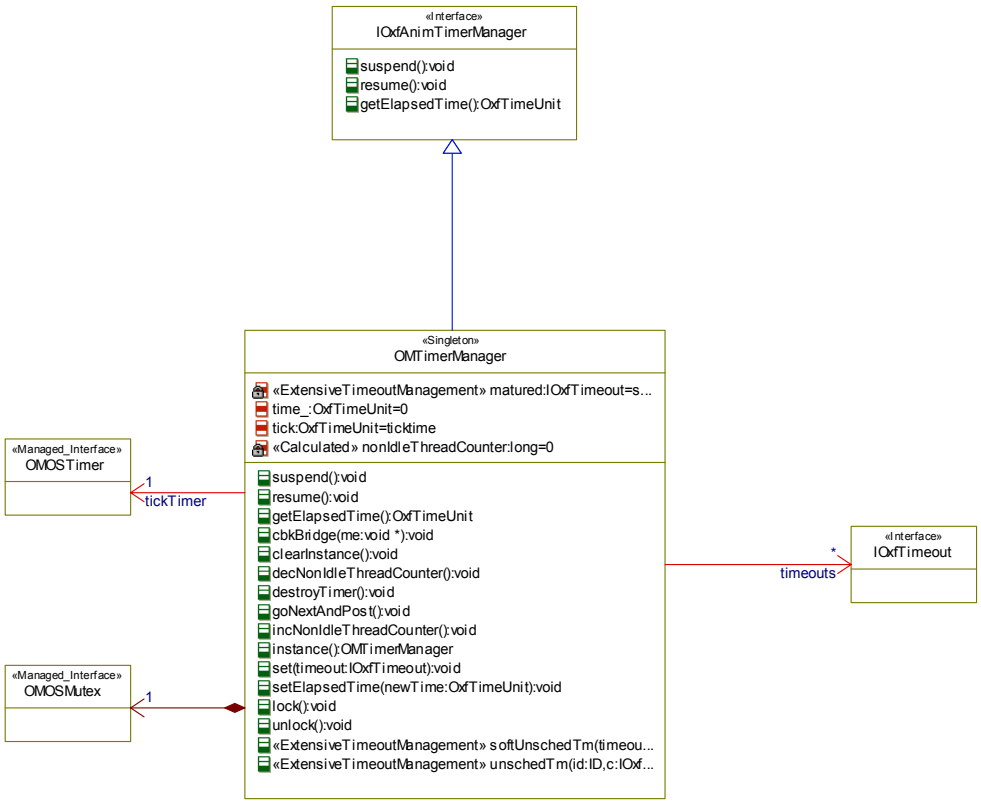
Description: A callback function signature that should be passed to the tick-timer to perform a tick action
Kind: Language
Declaration: typedef void (*%s)(void*);

Package: TimeManagement

Object Model Diagram Information

Object Model Diagram name: The timer manager

Description: This diagram shows the timer manager main relationships



Class Information for Package: TimeManagement

Class name: OMDelay

Description:Delay the calling thread
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMDelay

Operation name: ~OMDelay

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: ~OMDelay()
Return Type:
Description: Cleanup

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: operator=(OMDelay delay)
Return Type: OMDelay
Description: Disabled assignment operator

Argument information for Operation operator=

Name	Type	Direction
delay	OMDelay	In

Operation name: wakeup

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: wakeup()
Return Type: void
Description: Wakeup the delayed thread

Operation name: OMDelay

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: OMDelay(OMDelay delay)
Return Type:
Description: the copy ctor and the assignment operator should not be used

Argument information for Operation OMDelay

Name	Type	Direction
delay	OMDelay	In

Operation name: OMDelay

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMDelay(OxfTimeUnit t)
Return Type:
Description: Initialize a delay class

Argument information for Operation OMDelay

Name	Type	Direction
t	OxfTimeUnit	In

Relation information for Class OMDelay

Relation name: stopSignal

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: stopSignal
LinkName:
RoleName: stopSignal
Type: Composition
Description:

Name	Inverse	Source	Target
stopSignal		OMDelay	OMOSEventFlag

Class name: OMTimerManagerDefaults

Description:Default values for the timer manager initialization
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMTimerManagerDefaults

Attribute Name: defaultMaxTM

Default Value: 100
Static: true
Visibility: public
Type: unsigned int
Stereotype:

Description: default value of the number of tm-s that can co-exist simultaneously (either on the heap or in the matured list)

Attribute Name: defaultTicktime

Default Value: 100
Static: true
Visibility: public
Type: unsigned int
Stereotype:
Description: default value of the timer's tick length

Class name: OMTimerManager

Description:The timer manager is responsible for timeout bookkeeping and dispatching.
In Extensive Timeout Management mode it is also responsible for timeouts canceling.
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Attribute Information for Class: OMTimerManager

Attribute Name: matured

Default Value: static_cast<int>(maxTM)
Static: false
Visibility: private
Type: IOxfTimeout
Stereotype:
Description: The matured timeouts

Attribute Name: time_

Default Value: 0
Static: false
Visibility: public
Type: OxfTimeUnit
Stereotype:
Description: The current system time

Attribute Name: tick

Default Value: ticktime
Static: false
Visibility: public
Type: OxfTimeUnit
Stereotype:
Description: timer resolution, updated every tick ms and counts time

Attribute Name: nonIdleThreadCounter

Default Value: 0
Static: false
Visibility: private
Type: long
Stereotype:
Description: The number of active threads.
Used for simulated time support (a tick occur only when all the threads are idle).

Attribute Name: realTimeModel

Default Value: true
Static: false
Visibility: public
Type: bool
Stereotype:
Description: time model can be real or simulated

Attribute Name: suspended

Default Value: false
Static: false
Visibility: private
Type: bool
Stereotype:
Description: Used by AOM to suspend/resume

Attribute Name: overflowMark

Default Value: 0x80000000
Static: true
Visibility: private
Type: OxfTimeUnit
Stereotype:
Description: overflow watermark;

Attribute Name: timerManagerSingletonDestroyed

Default Value: false
Static: true
Visibility: private
Type: bool
Stereotype:
Description: Singleton state flag, used to identify that the singleton is destroyed (due to exit())

Operation information for Class: OMTimerManager

Operation name: ~OMTimerManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: ~OMTimerManager()
Return Type:
Description: Cleanup

Operation name: OMTimerManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private

Signature: OMTimerManager(OMTimerManager other)
Return Type:
Description: disable copy CTOR and = operator

Argument information for Operation OMTimerManager

Name	Type	Direction
other	OMTimerManager	In

Operation name: OMTimerManager

Initializer: timeouts(static_cast<int>(maxTM))
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: OMTimerManager(OxfTimeUnit ticktime,unsigned int maxTM,bool isRealTimeModel)
Return Type:
Description: Initialize

Argument information for Operation OMTimerManager

Name	Type	Direction
ticktime	OxfTimeUnit	In
maxTM	unsigned int	In
isRealTimeModel	bool	In

Operation name: action

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: action(IOxfTimeout timeout)
Return Type: void
Description: Send a matured timeout to its destination.
Also wakeup completed delays.

Argument information for Operation action

Name	Type	Direction
timeout	IOxfTimeout	In

Operation name: cbkBridge

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public

Signature: cbkBridge(void * me)
Return Type: void
Description: the timer manager callback (activated by the tick-timer)

Argument information for Operation cbkBridge

Name	Type	Direction
me	void *	In

Operation name: clearInstance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: clearInstance()
Return Type: void
Description: Singleton cleanup

Operation name: decNonIdleThreadCounter

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: decNonIdleThreadCounter()
Return Type: void
Description: Reduce the number of active threads

Operation name: destroyTimer

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: destroyTimer()
Return Type: void
Description: Cleanup

Operation name: getStaticTimerManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: private
Signature: getStaticTimerManager()

Return Type: OMTimerManager
Description: return a static instance of the timer manager

Operation name: getStaticTimerManager

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: private
Signature: getStaticTimerManager(OxfTimeUnit tickTime,unsigned int maxTM,bool isRealTimeModel,bool forceInitialization)
Return Type: OMTimerManager
Description: Create/get the timer manager singleton

Argument information for Operation getStaticTimerManager

Name	Type	Direction
tickTime	OxfTimeUnit	In
maxTM	unsigned int	In
isRealTimeModel	bool	In
forceInitialization	bool	In

Operation name: goNext

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: goNext()
Return Type: void
Description: simulated time/instrumentation tick

Operation name: goNextAndPost

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: goNextAndPost()
Return Type: void
Description: Advance the simulated/instrumentation time and send matured timeouts

Operation name: incNonIdleThreadCounter

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: public
Signature: incNonIdleThreadCounter()
Return Type: void
Description: Increase the number of active threads

Operation name: init

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: init()
Return Type: void
Description: Initialize the timer manager

Operation name: initInstance

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: initInstance(OxfTimeUnit ticktime,unsigned int maxTM,bool isRealTimeModel)
Return Type: OMTimerManager
Description: Lazy initialization of the timer manager singleton

Argument information for Operation initInstance

Name	Type	Direction
ticktime	OxfTimeUnit	In
maxTM	unsigned int	In
isRealTimeModel	bool	In

Operation name: initTimeoutsMemoryPool

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: private
Signature: initTimeoutsMemoryPool()
Return Type: void
Description: initialize the timeouts static memory pool

Operation name: instance

Initializer:
Const: false
Trigger: false
Abstract: false

Static: true
Virtual: false
Visibility: public
Signature: instance()
Return Type: OMTimerManager
Description: Get the singleton

Operation name: operator=

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: operator=(OMTimerManager other)
Return Type: OMTimerManager
Description: Disabled assignment operator

Argument information for Operation operator=

Name	Type	Direction
other	OMTimerManager	In

Operation name: post

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: post()
Return Type: void
Description: handle the matured timeouts, and handle timer overflow

Operation name: resume

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: resume()
Return Type: void
Description: Design level debugging support - resume time processing

Operation name: set

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false

Virtual: false
Visibility: public
Signature: set(IOxfTimeout timeout)
Return Type: void
Description: set - adding a timeout to be managed

Argument information for Operation set

Name	Type	Direction
timeout	IOxfTimeout	In

Operation name: setElapsedTime

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: setElapsedTime(OxfTimeUnit newTime)
Return Type: void
Description: Update the time

Argument information for Operation setElapsedTime

Name	Type	Direction
newTime	OxfTimeUnit	In

Operation name: suspend

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: suspend()
Return Type: void
Description: Design level debugging support - suspend time processing

Operation name: timeTickCbk

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: timeTickCbk()
Return Type: void
Description: respond to a tick

Operation name: resetTimeoutsDueTime

Initializer:

Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: resetTimeoutsDueTime()
Return Type: void
Description: Correct the timeouts due time and the time itself when the time_ field overflows.

Operation name: setTimeoutDueTime

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: setTimeoutDueTime(IOxfTimeout timeout)
Return Type: void
Description: Set the timeout due time.
Done when the timeout is added to the manager based on the timeout delay and the current system time.

Argument information for Operation setTimeoutDueTime

Name	Type	Direction
timeout	IOxfTimeout	In

Operation name: lock

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: lock()
Return Type: void
Description: Start a guarded critical section

Operation name: unlock

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: unlock()
Return Type: void
Description: End of the manager critical section

Operation name: findInList

Initializer:

Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: private
Signature: findInList(ID id,IOxfReactive c)
Return Type: IOxfTimeout
Description: find a timeout by id & destination (for timeout cancellation, on reactive class deletion)

Argument information for Operation findInList

Name	Type	Direction
id	ID	In
c	IOxfReactive	In

Operation name: softUnschedTm

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: softUnschedTm(OMTimeout timeout)
Return Type: void
Description: unschedule the timeout - used only from ~Timeout()

Argument information for Operation softUnschedTm

Name	Type	Direction
timeout	OMTimeout	In

Operation name: unschedTm

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: unschedTm(ID id,IOxfReactive c)
Return Type: void
Description: Canceling a timeout for a single object or a single object.

This method is used
(1) in case of exiting a state - timeout no longer relevant
(2) In case where the object is destroyed. In that case all timers associated with the object are destroyed.

Canceling a timeout requires one of two actions:
- Deleting the timeout from the timeouts, or
- Canceling it inside the event queue, if already dispatched. This in done by iterating the "enqueued but not yet dispatched" list.

Argument information for Operation unschedTm

Name	Type	Direction
id	ID	In
c	IOxfReactive	In

Operation name: action

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: protected
Signature: action(OMTimeout timeout)
Return Type: void
Description: Backward compatibility API.
Send a matured timeout to its destination.
Also wakeup completed delays.

Argument information for Operation action

Name	Type	Direction
timeout	OMTimeout	In

Operation name: getElapsedTime

Initializer:
Const: true
Trigger: false
Abstract: false
Static: false
Virtual: false
Visibility: public
Signature: getElapsedTime()
Return Type: OxfTimeUnit
Description: Returns the elapsed time

Operation name: advanceTime

Initializer:
Const: false
Trigger: false
Abstract: false
Static: false
Virtual: true
Visibility: public
Signature: advanceTime()
Return Type: void
Description: advance the system time to the next waiting timeout

Generalization information for Class OMTimerManager

Generalization name: IOxfAnimTimerManager

Description:

Virtual: false
Visibility: public
Extension Point:

Name	Base	Derived
IOxfAnimTimerManager	IOxfAnimTimerManager	OMTimerManager

Relation information for Class OMTimerManager

Relation name: timeouts

Symmetric: false
Multiplicity: *
Qualifier:
Visibility: public
Label: timeouts
LinkName:
RoleName: timeouts
Type: Association
Description: The pending timeouts (not yet matured)

Relation name: tickTimer

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: tickTimer
LinkName:
RoleName: tickTimer
Type: Association
Description:

Relation name: guard

Symmetric: false
Multiplicity: 1
Qualifier:
Visibility: public
Label: guard
LinkName:
RoleName: guard
Type: Composition
Description:

Name	Inverse	Source	Target
timeouts		OMTimerManager	IOxfTimeout
tickTimer		OMTimerManager	OMOSTimer
guard		OMTimerManager	OMOSMutex

Type information for Package TimeManagement

Type name: OMAbstractTickTimerFactory

Description: typedef of IOxfTickTimerFactory for backward compatibility
Kind: Typedef
Basic Type: IOxfTickTimerFactory

Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Operation information for Package TimeManagement

Operation name: operator >

Static: false
Signature: operator >(IOxfTimeout t1,IOxfTimeout t2)
Description: Greater than timeouts compare that compares two timeouts based on the due time.
Used to sort & find the timeouts collection of the timer manager.

Operation name: operator <

Static: false
Signature: operator <(IOxfTimeout t1,IOxfTimeout t2)
Description: Less than timeouts compare that compares two timeouts based on the due time.
Used to sort & find the timeouts collection of the timer manager.

Operation name: operator ==

Static: false
Signature: operator ==(IOxfTimeout t1,IOxfTimeout t2)
Description: Equal to timeouts compare that compares two timeouts based on the due time.
Used to sort & find the timeouts collection of the timer manager.

Package: Types

Package Information

Description: Basic types

Package: BasicTypes

Type information for Package BasicTypes

Type name: OMAPPLICATION

Description: OMAPPLICATION indicates that the framework is used by a client application.
OMOMATE indicates that the framework is used by Rhapsody.
Kind: Language
Declaration: #ifndef OMOMATE
// application
#ifndef %s
#define %s
#endif
#else

// check that there is no multiple definition
#ifdef %s
// Protect against both flags 'on'
#error "'OMOMATE' and 'OMAPPLICATION' cannot be defined together"
#endif // %s
#endif // OMOMATE

Type name: IncludeOMString

Description: Selective include to OMString
Kind: Language
Declaration: #ifdef OMAPPLICATION
#include "omstring.h"
#endif // OMAPPLICATION

Type name: RhpBoolean

Description: Language independent boolean type supported by Rhapsody.
Kind: Typedef
Basic Type: bool
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: RhpCharacter

Description: Language independent character type supported by Rhapsody.
Kind: Typedef
Basic Type: char
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: RhpAddress

Description: Language independent address type supported by Rhapsody.
Kind: Typedef
Basic Type: void *
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: RhpInteger

Description: Language independent integer type supported by Rhapsody.
Kind: Typedef
Basic Type: int
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: RhpPositive

Description: Language independent positive integer type supported by Rhapsody.
Kind: Typedef
Basic Type: unsigned int

Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: RhpReal

Description: Language independent real number type supported by Rhapsody.
Kind: Typedef
Basic Type: double
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: RhpString

Description: Language independent string type supported by Rhapsody.
Kind: Typedef
Basic Type: OMString
Multiplicity: 1
Constant: false
Reference: true
Ordered: false

Type name: RhpUnlimitedNatural

Description: Language independent natural number type supported by Rhapsody.
Kind: Typedef
Basic Type: long
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: RhpVoid

Description: Language independent VOID type supported by Rhapsody.
Kind: Typedef
Basic Type: void
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: OMBoolean

Description: Boolean type
Kind: Typedef
Basic Type: RhpBoolean
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: FALSE

Description: OMBoolean false value
Kind: Language
Declaration: #ifndef %s

#define %s false
#endif // %s

Type name: TRUE

Description: OMBoolean true value
Kind: Language
Declaration: #ifndef %s
#define %s true
#endif // %s

Type name: OMitoa

Description: OMitoa declaration
Kind: Language
Declaration: #ifndef %s
extern void %s(int val, char* str, int base = 10);
#endif // %s

Type name: TMPL_INL

Description: Compilation dependent addition of the inline keyword in the definition of template classes operations
Kind: Language
Declaration: #ifdef NEED_INLINE_IN_TEMPLATE
#define TMPL_INL inline
#else
#define TMPL_INL
#endif // NEED_INLINE_IN_TEMPLATE

Type name: OMHandle

Description: OMHandles - the "names" of various model objects.
Kind: Typedef
Basic Type: char
Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: OMRAW_MEMORY_ALIGNMENT

Description: default memory alignment
Kind: Language
Declaration: #ifndef %s
#define %s 8
#endif // %s

Type name: USE_DYNAMIC_MEMORY_ALLOCATION

Description: dynamic memory allocation exists (new/delete)
Kind: Language
Declaration: #ifndef OM_NO_DYNAMIC_MEMORY_ALLOCATION
#define %s
#endif // OM_NO_DYNAMIC_MEMORY_ALLOCATION

Type name: timeUnit

Description: A TimeUnit compatibility name
Kind: Typedef
Basic Type: OxfTimeUnit

Multiplicity: 1
Constant: false
Reference: false
Ordered: false

Type name: OMDefaultThread

Description: A reactive object default active context
Kind: Language
Declaration: #define %s NULL

Type name: _OMINSTRUMENT

Description: set the _OMINSTRUMENT flag
Kind: Language
Declaration: #if (defined OMTRACER || defined OMANIMATOR)
#ifndef %s
#define %s
#endif // %s
#endif // (defined OMTRACER || defined OMANIMATOR)

Type name: aomArg

Description: AOM Argument declaration
Kind: Language
Declaration: #ifdef _OMINSTRUMENT
#define %s(arg) arg
#else
#define %s(arg)
#endif // _OMINSTRUMENT

Type name: omtypename

Description: Define omtypename to allow environment dependent usage of the typename keyword
Kind: Language
Declaration: #ifndef OM_NO_TYPENAME_SUPPORT
#define %s typename
#else
// avoid the typename keyword
#define %s
#endif // OM_NO_TYPENAME_SUPPORT

Type name: OM_NO_THROW

Description: Definition of OM_NO_THROW that is translated to throw() when
OM_NEED_THORW_IN_NEW_OPERATOR is defined and to nothing otherwise.
Kind: Language
Declaration: #ifdef OM_NEED_THORW_IN_NEW_OPERATOR
#define %s throw()
#else
#define %s
#endif // OM_NEED_THORW_IN_NEW_OPERATOR

Package: omiotypes

Type information for Package omiotypes

Type name: iosfwd

Description: Forward declaration to STL io types
Kind: Language
Declaration: #if ((defined OM_STL) || (defined OM_USE_STL))
#include <iosfwd>
#endif // ((defined OM_STL) || (defined OM_USE_STL))

Type name: STD_NAMESPACE

Description: Definition of STD_NAMESPACE when it is not defined by the include to STLMacros
Kind: Language
Declaration: #ifndef %s
#define %s
#endif // %s

Type name: omcerr

Description: cerr definition
Kind: Language
Declaration: #define %s STD_NAMESPACE cerr

Type name: omcin

Description: cin definition
Kind: Language
Declaration: #define %s STD_NAMESPACE cin

Type name: omcout

Description: cout definition
Kind: Language
Declaration: #define %s STD_NAMESPACE cout

Type name: omendl

Description: endl definition
Kind: Language
Declaration: #define %s STD_NAMESPACE endl

Type name: omends

Description: ends definition
Kind: Language
Declaration: #define %s STD_NAMESPACE ends

Type name: omflush

Description: flush definition
Kind: Language
Declaration: #define %s STD_NAMESPACE flush

Type name: omhex

Description: hex definition
Kind: Language
Declaration: #define %s STD_NAMESPACE hex

Type name: omifstream

Description: ifstream definition
Kind: Language
Declaration: #define %s STD_NAMESPACE ifstream

Type name: omistream

Description: istream definition
Kind: Language
Declaration: #define %s STD_NAMESPACE istream

Type name: omistrstream

Description: istrstream definition
Kind: Language
Declaration: #define %s STD_NAMESPACE istrstream

Type name: omofstream

Description: ofstream definition
Kind: Language
Declaration: #define %s STD_NAMESPACE ofstream

Type name: omostream

Description: istream definition
Kind: Language
Declaration: #define %s STD_NAMESPACE ostream

Type name: omostrstream

Description: ostrstream definition
Kind: Language
Declaration: #define %s STD_NAMESPACE ostrstream

Package: Unicode

Class Information for Package: Unicode

Class name: OMUnicodeHelper

Description:A utility class for unicode resolution
Active: false
Behavior Overridden: false
Composite: false
Reactive: false

Operation information for Class: OMUnicodeHelper

Operation name: toupper

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: toupper(int c)
Return Type: int
Description: Returns the c character, converted to upper case
If the c is already uppercased, it returns the same value

Argument information for Operation toupper

Name	Type	Direction
c	int	In

Operation name: wtoc

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: wtoc(char* cstr,wchar_t wcstr,size_t count)
Return Type: size_t
Description: converts the wcstr wide character string to char* string
returns size of cstr

Argument information for Operation wtoc

Name	Type	Direction
cstr	char*	Out
wcstr	wchar_t	In
count	size_t	In

Operation name: ctow

Initializer:
Const: false
Trigger: false
Abstract: false
Static: true
Virtual: false
Visibility: public
Signature: ctow(wchar_t wcstr,char* cstr,size_t count)
Return Type: size_t
Description: converts the char* string to the wide character string
returns size of the wide character string

Argument information for Operation ctow

Name	Type	Direction
wcstr	wchar_t	Out
cstr	char*	In
count	size_t	In

Type information for Package Unicode

Type name: OMctow

Description: OMUnicodeHelper::ctow() compatibility name
Kind: Language
Declaration: #define %s OMUnicodeHelper::ctow

//

Type name: OMtoupper

Description: OMUnicodeHelper::toupper() compatibility name
Kind: Language
Declaration: #define %s OMUnicodeHelper::toupper
//

Type name: OMwtoc

Description: OMUnicodeHelper::wtoc() compatibility name
Kind: Language
Declaration: #define %s OMUnicodeHelper::wtoc
//

Package: StandardTypes

Description: Standard ANSI types

Type information for Package StandardTypes

Type name: size_t

Description: ANSI C size type
Kind: Language

Type name: time_t

Description: ANSI C time type
Kind: Language

Type name: wchar_t

Description: ANSI C wide-char type
Kind: Language

Components Information

Component Name:aom

Type:library
Directory:L:\Programs\Rhapsody_Aquarius\Share\LangCpp\oxf\model\..\..
Libraries:
Additional Sources:
Standard Headers:
Include Path:
Description:The external AOM component

File information for Component: aom

Files

Path:..
File Type:folder

Description:

File information for Files

aom

Path:
File Type:folder
Description:

Configuration information for Component: aom

generic Configuration

Configuration Name: generic
Description: The AOM configuration related to the OXF
Initialization Scope:explicit
Initialization Code:
Directory:..\..
Libraries:
Additional Sources:
Standard Headers
Include Path:
Instrumentation:none
Time Model:real
Statechart Implementation:flat
BuildSet: Debug
Compiler Switches: /I . /I $OMDefaultSpecificationDirectory /I $(OMROOT)\LangCpp /I $(OMROOT)\LangCpp\oxf /nologo /W3 /GX $OMCPPCompileCommandSet /D "_AFXDLL" /D "WIN32" /D "_CONSOLE" /D "_MBCS" /D "_WINDOWS" $(INST_FLAGS) $(INCLUDE_PATH) $(INST_INCLUDES) /c
Link Switches:$OMLinkCommandSet /NOLOGO

Component Name:oxfAnimFiles

Type:library
Directory:L:\Programs\Rhapsody_Aquarius\Share\LangCpp\oxf\model\oxfAnimFiles\../..
Libraries:
Additional Sources:
Standard Headers:
Include Path:
Description:Animation required files and services.
It is assumed that when using this subset of the OXF one will define the OM_NO_FRAMEWORK_MEMORY_MANAGER compilation flag.

File information for Component: oxfAnimFiles

Files

Path:
File Type:folder
Description:

File information for Files

omstring
Path:
File Type:logical
Description:OMString

rawtypes
Path:
File Type:specification
Description:Basic types

os
Path:
File Type:specification
Description:The OSAL interface

rp_framework_dll_definition
Path:
File Type:specification
Description:DLL macros definition

EMPTY_IMPLEMENTATION
Path:
File Type:implementation
Description:This file is used to avoid the generation of empty implementation files (of the mapped elements)

omlist
Path:
File Type:specification
Description:OMList<>

ommap
Path:
File Type:specification
Description:OMMap<>

ommemorymanager
Path:
File Type:specification
Description:The memory manager

omprotected
Path:
File Type:logical
Description:OMProtected and OMGuard

omqueue
Path:
File Type:specification
Description:OMQueue<>

omstack

Path:
File Type:specification
Description:OMStack<>

os

Path:
File Type:specification
Description:The OSAL interface

omiotypes

Path:
File Type:specification
Description:iostream types

OXFSelectiveInclude

Path:
File Type:specification
Description:selective (#ifdef based) include statements

OMAbstractContainer

Path:
File Type:specification
Description:The OM Collections abstract container type

OMNullValue

Path:
File Type:specification
Description:A container NULL value return type

OMIterator

Path:
File Type:specification
Description:Iterator on OMAbstractContainer

OXFGuardMacros

Path:
File Type:specification
Description:OMProtected guard macros

OMResourceGuard

Path:
File Type:specification
Description:The resource guard template class

omcollec

Path:
File Type:specification
Description:OMCollection - dynamic array

OMStaticArray

Path:
File Type:specification

Description:Static size array

OXFNotifyMacros

Path:
File Type:specification
Description:Notify macros

OMNotifier

Path:
File Type:logical
Description:Notifier class

omtypes

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

omunicode

Path:
File Type:specification
Description:Unicode support

OXFMemoryManagerMacros

Path:
File Type:specification
Description:memory manager macros

OXFManager

Path:
File Type:specification
Description:memory management

IOxfMemoryAllocator

Path:
File Type:specification
Description:memory manager API

Configuration information for Component: oxfAnimFiles

generic Configuration

Configuration Name: generic
Description: The oxf generic framework configuration - animation subset.
Generates the elements and the oxfAnimFiles.list
Initialization Scope:explicit
Initialization Code:
Directory:oxfAnimFiles\../..
Libraries:
Additional Sources:
Standard Headers
Include Path:

Instrumentation:none
Time Model:real
Statechart Implementation:flat
BuildSet: NA
Compiler Switches:
Link Switches:

Component Name:oxfFiles

Type:library
Directory:L:\Programs\Rhapsody_Aquarius\Share\LangCpp\oxf\model\oxfFiles\../..
Libraries:
Additional Sources:
Standard Headers:
Include Path:
Description:The generic framework interfaces and implementation

File information for Component: oxfFiles

Files

Path:
File Type:folder
Description:

File information for Files

omstring

Path:
File Type:logical
Description:OMString

rawtypes

Path:
File Type:specification
Description:Basic types

os

Path:
File Type:specification
Description:The OSAL interface

rp_framework_dll_definition

Path:
File Type:specification
Description:DLL macros definition

OMObsolete

Path:
File Type:specification
Description:Obsolete names compatibility

omcollec

Path:
File Type:specification
Description:OMCollection<>

event

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

timer

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

omtypes

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

HdlCls

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

omoutput

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

state

Path:
File Type:logical
Description:Reusable statechart base state classes

AMemAloc

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

MemAlloc

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

EMPTY_IMPLEMENTATION

Path:
File Type:implementation
Description:This file is used to avoid the generation of empty implementation files (of the mapped elements)

omheap

Path:
File Type:specification
Description:OMHeap<>

omlist

Path:
File Type:specification
Description:OMList<>

ommap

Path:
File Type:specification
Description:OMMap<>

ommemorymanager

Path:
File Type:logical
Description:The memory manager

omprotected

Path:
File Type:logical
Description:OMProtected and OMGuard

omqueue

Path:
File Type:specification
Description:OMQueue<>

omreactive

Path:
File Type:logical
Description:OMReactive

omstack

Path:
File Type:specification
Description:OMStack<>

omthread

Path:
File Type:logical
Description:OMThread

omucollec

Path:
File Type:specification
Description:OMUCollection

omulist

Path:
File Type:specification
Description:OMUList

omumap

Path:
File Type:specification
Description:OMUMap

oxf

Path:
File Type:logical
Description:The OXF utility class

IOxfMemoryAllocator

Path:
File Type:specification
Description:The IOxfMemoryAllocator interface

omstatic

Path:
File Type:specification
Description:Rhapsody 5.X compatibility file
This file is used to include files of elements that were located in this file in Rhapsody 5.X.
The element file was renamed to improve the naming.

os

Path:
File Type:specification
Description:The OSAL interface

Configuration information for Component: oxfFiles

generic Configuration

Configuration Name: generic
Description: The oxf generic framework configuration.
Generates the elements and the oxfFiles.list
Initialization Scope:explicit
Initialization Code:
Directory:oxfFiles\../..

Libraries:
Additional Sources:
Standard Headers
Include Path:
Instrumentation:none
Time Model:real
Statechart Implementation:flat
BuildSet: NA
Compiler Switches:
Link Switches:

generic_dll Configuration

Configuration Name: generic_dll
Description: This configuration is designed to generate the compilation rules for the DLL version of the OXF.
It should be used only to generate its makefile.
Actual code generation should be done via the generic configuration.
Initialization Scope:explicit
Initialization Code:
Directory:oxfFiles\../..
Libraries:
Additional Sources:
Standard Headers
Include Path:
Instrumentation:none
Time Model:real
Statechart Implementation:flat
BuildSet: NA
Compiler Switches:
Link Switches: