

Actor Framework

Antidoc v3.0.0, Ron Dexter

Table of Contents

1. Project description	1
2. Libraries	2
2.1. Actor Framework.lvlib	2
2.2. AF Debug.lvlib	3
3. Classes	6
3.1. Classes overview	6
3.2. Message.lvclass	6
3.3. Stop Msg.lvclass	8
3.4. Batch Msg.lvclass	10
3.5. Reply Msg.lvclass	11
3.6. Report Error Msg.lvclass	13
3.7. Last Ack.lvclass	14
3.8. Launch Nested Actor Msg.lvclass	16
3.9. Message Priority Queue.lvclass	17
3.10. Message Enqueuer.lvclass	18
3.11. Message Dequeuer.lvclass	19
3.12. Message Queue.lvclass	21
3.13. Protected Actor Handle.lvclass	23
3.14. Ping Msg.lvclass	25
3.15. Register Actor Msg.lvclass	27
4. Actors (AF)	29
4.1. Preamble	29
4.2. Actors overview	29
4.3. Actor.lvclass	29
5. Legal Information	38
5.1. Document creation	38
5.2. Product used in the project	40

Chapter 1. Project description

No description found (add content in project description)

Chapter 2. Libraries

This section describes the libraries contained in the project.

2.1. Actor Framework.lvlib

Responsibility: No description found (add content in lvlib description)

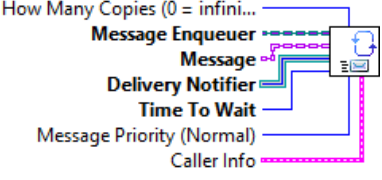

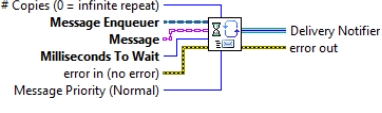


Version: 4.1.0.0

Table 1. Nested libraries

Name	Type
Message.lvclass	LVClass
Stop Msg.lvclass	LVClass
Batch Msg.lvclass	LVClass
Reply Msg.lvclass	LVClass
Report Error Msg.lvclass	LVClass
Last Ack.lvclass	LVClass
Launch Nested Actor Msg.lvclass	LVClass
Actor.lvclass	LVClass
Message Priority Queue.lvclass	LVClass
Message Enqueuer.lvclass	LVClass
Message Dequeuer.lvclass	LVClass
Message Queue.lvclass	LVClass
Protected Actor Handle.lvclass	LVClass
Ping Msg.lvclass	LVClass
Register Actor Msg.lvclass	LVClass
AF Debug.lvlib	Library

2.1.1. Functions

Table 2. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Time-Delayed Send Message Core		<p>This VI forms the core processing loop for the time-delayed messages. This VI may be called directly if the desired behavior is for the caller to wait until the messages have been sent before continuing, but the usual usage of this VI is to call Time-Delayed Send Message.vi which will launch an asynch clone of this VI so that the caller can proceed with other work instead of waiting for the time out.</p> <p>See Context Help for Time-Delayed Send Message.vi for further details.</p>			
Time-Delayed Send Message		<p><p>Waits for the specified amount of time and then sends a message to an actor a specified number of times.</p></p>			
Init Actor Queues FOR TESTING ONLY		<p><p>Provides access the To-Self and To-Caller message queues for an actor without launching the actor.</p></p> <p><p>Use this VI to test how an actor handles messages. Do not use this VI in code that you deploy.</p></p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

2.1.2. Library Constant VIs

NOTE | No Constant VIs Found

2.2. AF Debug.lvlib

Responsibility: No description found (add content in lvlib description)

Version: 1.0.0.0



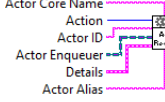


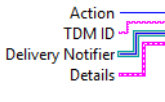

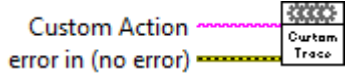

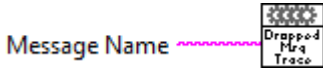
Table 3. Nested libraries

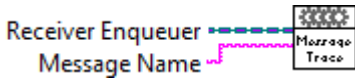
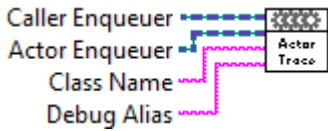
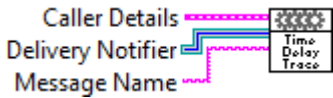
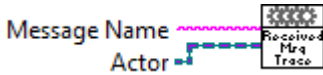
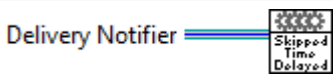
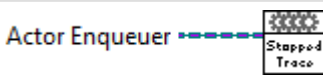
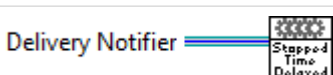
Name	Type
Protected Actor Handle.lvclass	LVClass
Ping Msg.lvclass	LVClass

Name	Type
Register Actor Msg.lvclass	LVClass

2.2.1. Functions

Table 4. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Get Actor Handles	 Actor Handles	This VI returns an array of handles for currently running actors to use in a debugging tool. Actors are wrapped in handles to limit a debugger's ability to interfere with the operation of the actors and to protect the debugger from actors stopping unexpectedly.			
Get Registry Update Event	 Event Registration Refnum	This VI returns an Event registration refnum for an Actor Framework debugger to use to hear about changes to the available debug information of actors. Every time this event fires, new information from the database is available for the debugger to harvest and display. The event itself does not carry any information to avoid duplicating information in the Event queue.			
Actor Registry	 Actor Core Name Action Actor ID Actor Enqueuer Details Actor Alias Event Registration Refnum Found This Actor All Actors Actor Alias out	No description found (add content in vi description)			
Get Clone Name	 Clone Type Clone ID	No description found (add content in vi description)			
TDM Registry	 Action TDM ID Delivery Notifier Details Found TDM ID out Delivery Notifier Details out TDM Alias Out	No description found (add content in vi description)			
Generate Custom Trace	 Custom Action error in (no error)	<p>Generates a DETT (Desktop Execution Trace Toolkit) User Generated Trace. The trace includes the source actor's ID (or debug alias, if available) and a custom message you define.</p>			
Generate Trace for Dropped Message	 Message Name	Generates a DETT User Generated Trace for a message object that was sent to an actor but never received because the actor stopped while the message was still in its queue.			

Name	Connector pane	Description	S.	R.	I.
Generate Trace for Message		Generates a DETT User Generated Trace for a message object being sent.			
Generate Trace for New Actor		Generates a DETT User Generated Trace for an actor being newly launched.			
Generate Trace for New Time Delayed Message		Generates a DETT User Generated Trace for a time-delayed message being created.			
Generate Trace for Received Message		Generates a DETT User Generated Trace for a message object being received by an actor.			
Generate Trace for Skipped Time-Delayed Message		Generates a DETT User Generated Trace for a repeated time-delayed message deliberately skipping one of its instances.			
Generate Trace for Stopped Actor		Generates a DETT User Generated Trace for an actor stopping, whether in response to the Stop Msg or an error or its normal termination.			
Generate Trace for Stopped Time-Delayed Message		Generates a DETT User Generated Trace for a time-delayed message transmitting its final iteration.			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

2.2.2. Library Constant VIs

NOTE | No Constant VIs Found

Chapter 3. Classes

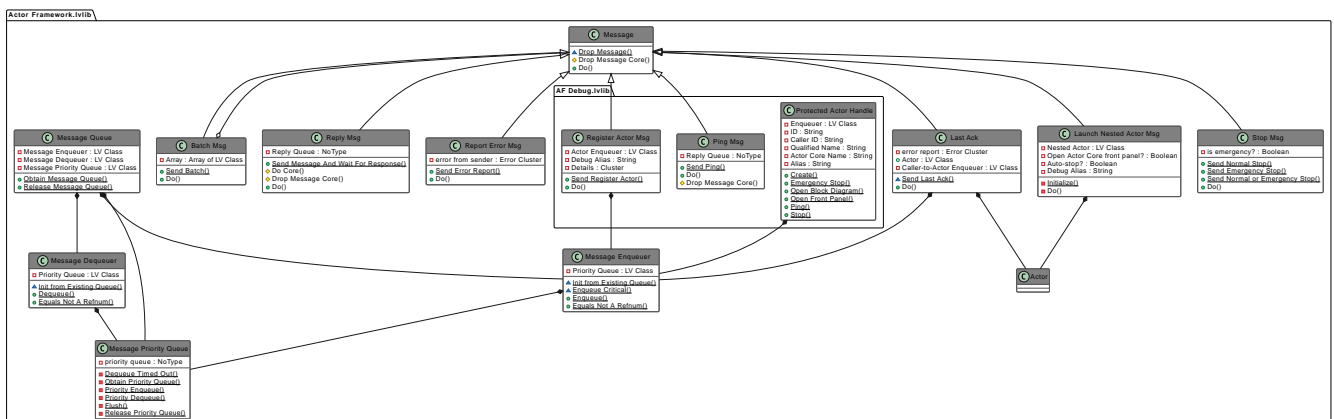
This section describes the classes contained in the project.

3.1. Classes overview

This project contains 14 classes and 0 interface.

Table 5. Classes list

Classes	Interfaces
Message.lvclass	
Stop Msg.lvclass	
Batch Msg.lvclass	
Reply Msg.lvclass	
Report Error Msg.lvclass	
Last Ack.lvclass	
Launch Nested Actor Msg.lvclass	
Message Priority Queue.lvclass	
Message Enqueuer.lvclass	
Message Dequeuer.lvclass	
Message Queue.lvclass	
Protected Actor Handle.lvclass	
Ping Msg.lvclass	
Register Actor Msg.lvclass	



3.2. Message.lvclass

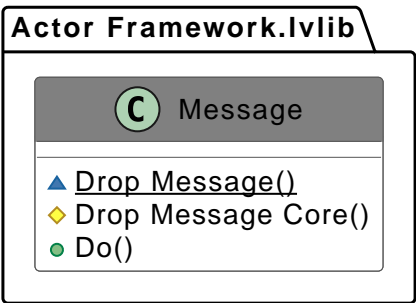
Responsibility: Message: <Any Actor> to <Any Actor>

This is the ancestor of all messages. Messages are sent via directed queue from the caller to the

actor or via a separate queue from the actor to the caller. In general, messages should be events along the lines of "you need to know this." They should not, generally, be synchronous requests for information of any kind. For further discussion of this, see comments on "Reply Message.lvclass".








Version: 1.0.0.0

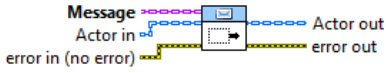

3.2.1. Diagram



3.2.2. Methods

Table 6. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Drop Message	Message in 	(Filename: Actor Framework.lvlib:Message.lvclass:Drop Message.vi) Wrapper for Drop Message Core.vi			
Drop Message Core	Message in 	(Filename: Actor Framework.lvlib:Message.lvclass:Drop Message Core.vi) Defines what a message does if it is in the message queue when the queue is released. At this point, the actor has shut down, so it will never process the message. By default, this method does nothing. A child class may override it to define behavior. National Instruments provides the Reply Msg class to override this method. This class releases its internal queue, which causes any caller of the Send Message and Wait For Response method to stop waiting and return an error.			

Name	Connector pane	Description	S.	R.	I.
Do		(Filename: Actor Framework.lvlib:Message.lvclass:Do.vi) Defines what a message does when it is received by an actor. Generally, a message instructs an actor to invoke one of its methods. By default, this method does nothing. A child class must override it to define behavior.			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.2.3. Class Constant VIs

NOTE | No Constant VIs Found

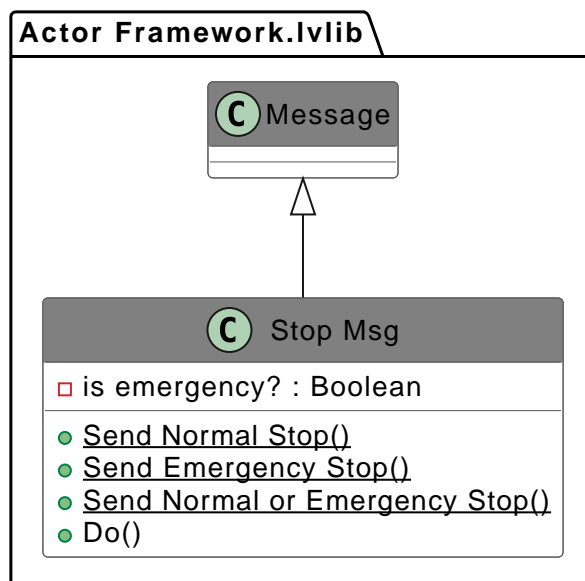
3.3. Stop Msg.lvclass

Responsibility: Message: Actor to Actor

When the Stop message is sent, it tells the actor to shutdown. The actor will stop handling further messages. The actor will send a Last Ack.lvclass message back on the queue to its caller. After sending the Stop message, the caller is free to exit and not wait for the Last Ack.

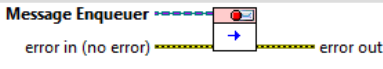


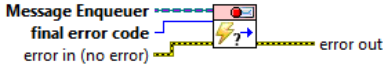

Version: 1.0.0.0

3.3.1. Diagram



3.3.2. Methods

Table 7. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Normal Stop		<p><p>(Filename: Actor Framework.lvlib:Stop Msg.lvclass:Send Normal Stop.vi)</p></p> <p><p>Sends a Stop message to an actor, triggering the actor to run its Stop Core method. This message has normal priority, meaning the actor will process this message after processing high- or normal-priority messages that are in the queue already.</p></p>			
Send Emergency Stop		<p><p>(Filename: Actor Framework.lvlib:Stop Msg.lvclass:Send Emergency Stop.vi)</p></p> <p><p>Sends an Emergency Stop message to an actor, triggering the actor to shut down as quickly as possible. This message has critical priority, meaning it will be processed before all messages that are already in the queue.</p></p>			
Send Normal or Emergency Stop		<p><p>(Filename: Actor Framework.lvlib:Stop Msg.lvclass:Send Normal or Emergency Stop.vi)</p></p> <p><p>Sends a Stop message to an actor. The priority of this message is determined by the value of the final error code input. If this input is 0, this VI sends the message with normal priority; otherwise, it sends the message with critical priority.</p></p>			
Do		No description found (add content in vi description)			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.3.3. Class Constant VIs

NOTE

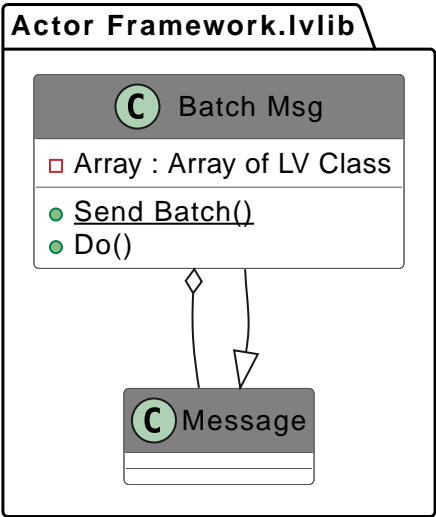
No Constant VIs Found

3.4. Batch Msg.lvclass

Responsibility: Use this class to aggregate many Messages together as a single Message so that they are all handled atomically, without other messages that may be enqueued from other sources getting interleaved.



Version: 1.0.0.0

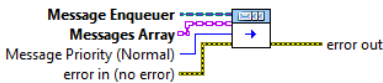

3.4.1. Diagram



3.4.2. Methods

Table 8. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Do		No description found (add content in vi description)			

Name	Connector pane	Description	S.	R.	I.
Send Batch		<p><p>(Filename: Actor Framework.lvlib:Batch Msg.lvclass:Send Batch.vi)</p></p> <p><p>Sends multiple messages in a batch to an actor. All messages in the batch are given the same priority. No additional messages of the same priority will be processed by the actor in between the messages in the batch. However, messages with a higher priority than the batch will be processed before the batch itself.</p></p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.4.3. Class Constant VIs

NOTE No Constant VIs Found

3.5. Reply Msg.lvclass

Responsibility: Message: <Any Actor> to ???

This is an abstract class. Children must override the Do.vi method.

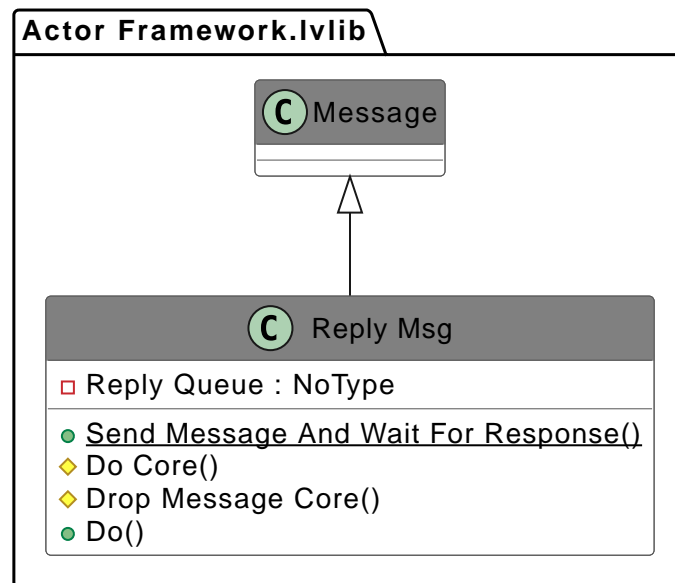
The Reply Message class is sent from caller to actor or from actor to caller. In either case, the sender of the message is going to wait for a synchronous reply from the recipient. The reply is sent using a private one-time-use queue.

Deadlock can occur if one party sends a message and starts waiting for a response and the other party at the same time sends a message and is waiting for a response. In general, avoid using this message class. You can generally redesign your application in one of two ways:

1) Instead of the first VI synchronously fetching data from the second VI, have the second VI send messages to the first VI any time the data in question gets updated. In that way, whenever the first VI needs to use a given value, it already has that value locally and does not need to synchronously fetch it. You may object that this means that there are two copies of the data in the system. This is true, but when you do the fetching of the data, you also create a duplicate. 2) Use a data value reference. This avoids the duplication of data but creates thread contention between the first and second VIs — one VIs execution can be hung waiting for the other VI to finish using the data value reference. Deadlock is also a risk with multiple data value references in the system.

Version: 1.0.0.0

3.5.1. Diagram



3.5.2. Methods

Table 9. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Do Core		<p>(Filename: Actor Framework.lvlib:Reply Msg.lvclass:Do Core.vi)</p> <p>Defines what an actor does when it receives Reply Message. This method also defines the Reply that is returned to the actor that sent Reply Message.</p> <p>By default, this method does nothing, and Reply is the ancestor Message class. A child class must override this method to define behavior and specify a child of the Message class to return in the Reply output.</p>	🔑	📄	
Drop Message Core		<p>Children of Message should override this method if they have behavior when the Actor receiving this message stops running before the message gets handled. Predominantly, this is overridden by Reply Message.lvclass in order to tell the waiting caller that there will never be a reply from the callee. This method probably does not need to be overridden by any other class.</p>	🔑	📄	
Do		No description found (add content in vi description)		📄	

Name	Connector pane	Description	S.	R.	I.
Send Message And Wait For Response		<p><p>(Filename: Actor Framework.lvlib:Reply Msg.lvclass:Send Message And Wait For Response.vi)</p></p> <p><p>Sends a message to an actor and synchronously waits for a response from the actor.</p></p>			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

3.5.3. Class Constant VIs

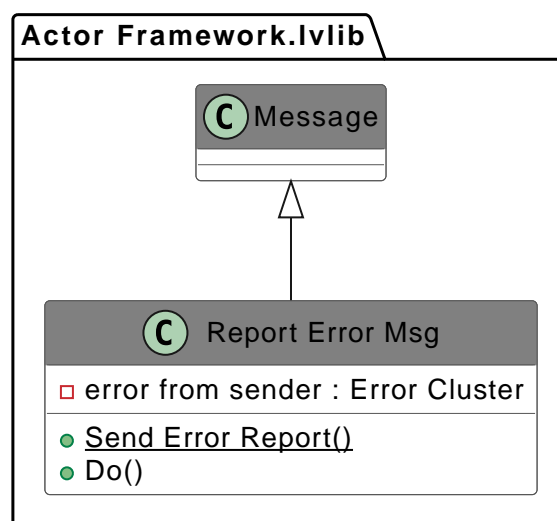
NOTE No Constant VIs Found

3.6. Report Error Msg.lvclass

Responsibility: Use this message class to send an error to an actor. The error will be handled by that actor in its **Handle Error Core.vi**. If the error is not handled there, it will cause the actor to stop running.



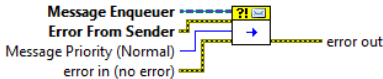


Version: 1.0.0.0

3.6.1. Diagram



3.6.2. Methods

Table 10. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Do		<p>(Filename: Actor Framework.lvlib:Message.lvclass:Do.vi)</p> <p>Defines what a message does when it is received by an actor. Generally, a message instructs an actor to invoke one of its methods.</p> <p>By default, this method does nothing. A child class must override it to define behavior. Create a child class by using the Actor Framework Message Maker dialog box.</p>			
Send Error Report		<p><p>(Filename: Actor Framework.lvlib:Report Error Msg.lvclass:Send Error Report.vi</p></p> <p><p>Use this VI to send an error to an actor. The error will be handled by the Handle Error VI of the actor. If the error is not handled there, it will cause the actor to stop running.</p></p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.6.3. Class Constant VIs

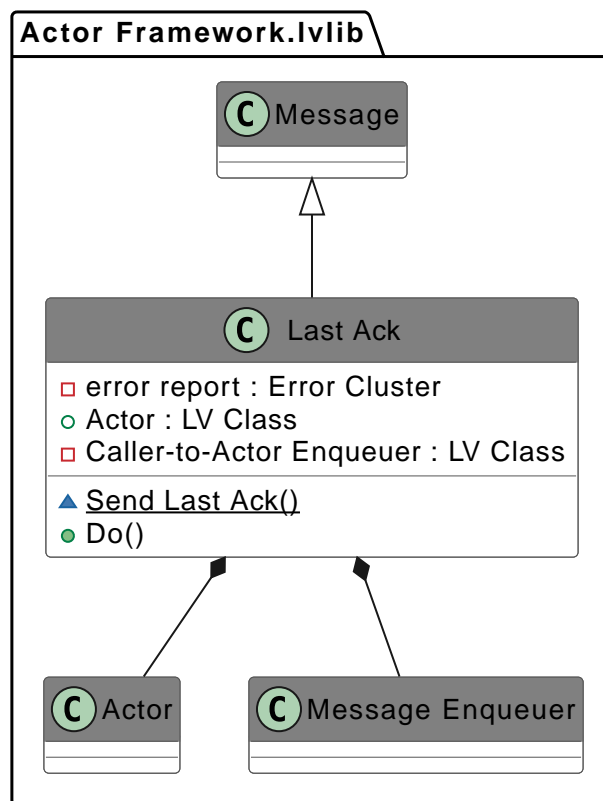
NOTE No Constant VIs Found

3.7. Last Ack.lvclass

Responsibility: Message: Actor to Actor

Last Ack stands for "Last Acknowledgement". It is a common term in network communication protocols signifying the very very last message when breaking a connection. In this case, a Last Ack message is sent from the actor to the caller whenever the actor shuts down. This shutdown may be because the actor has finished the work it was created to do, or because it generated an error, or because it received a Stop message. If it was because of an error, the error is recorded in the Last Ack. The final state of the actor as it exited is also recorded in the Last Ack. The caller may legitimately already have quit when the Last Ack is sent, so any error generated while sending the Last Ack is ignored.

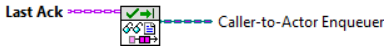
3.7.1. Diagram



3.7.2. Methods

Table 11. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Last Ack		No description found (add content in vi description)			
Do		No description found (add content in vi description)			
Read Actor		<p><p>(Filename: Actor Framework.lvlib:Last Ack.lvclass:Read Actor.vi)</p></p> <p><p>Returns the state the actor was in when it shut down.</p></p>			
Read Error Report		<p><p>(Filename: Actor Framework.lvlib:Last Ack.lvclass:Read Error Report.vi)</p></p> <p><p>Returns the error, if any, that caused an actor to shut down.</p></p>			

Name	Connector pane	Description	S.	R.	I.
Read Caller-To-Actor Enqueuer		<p><p>(Filename: Actor Framework.lvlib:Last Ack.lvclass:Read Caller-To-Actor Enqueuer.vi</p></p> <p><p>Returns the reference the caller used to send messages to the actor. Although the reference is invalid, you can use it to identify which actor shut down.</p></p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.7.3. Class Constant VIs

NOTE | No Constant VIs Found

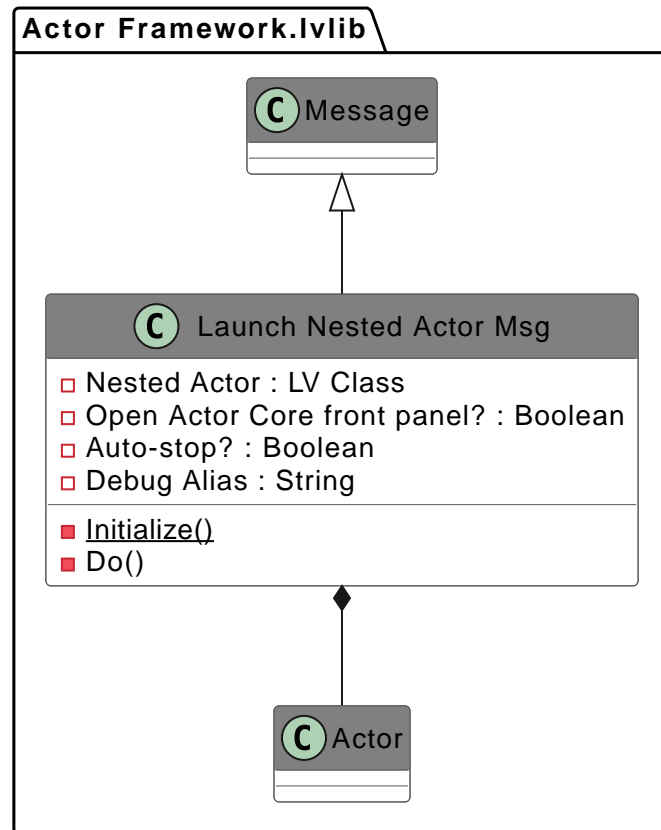
3.8. Launch Nested Actor Msg.lvclass

Responsibility: Message: Actor to Actor

Launch Nested Actor Msg is a private message of the Actor Framework. Unlike most message classes, it does not have a member VI for sending the message. The send for this message is instead a member of Actor.lvclass. For further information, please read the help for **Actor.lvclass:Send Launch Nested Actor Msg.vi**.

Version: 1.0.0.4

3.8.1. Diagram



3.8.2. Methods

This library has no functions set to non private scope.

3.8.3. Class Constant VIs

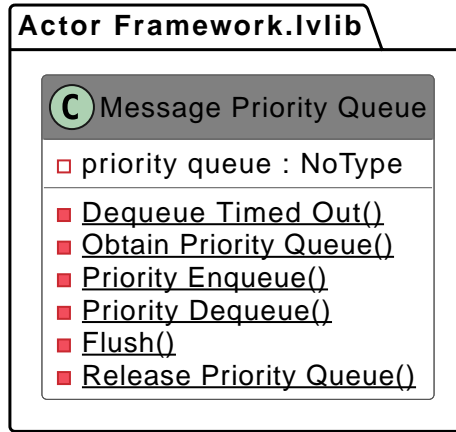
NOTE | No Constant VIs Found

3.9. Message Priority Queue.lvclass

Responsibility: No description found (add content in lvclass description)

Version: 1.0.0.1

3.9.1. Diagram



3.9.2. Methods

This library has no functions set to non private scope.

3.9.3. Class Constant VIs

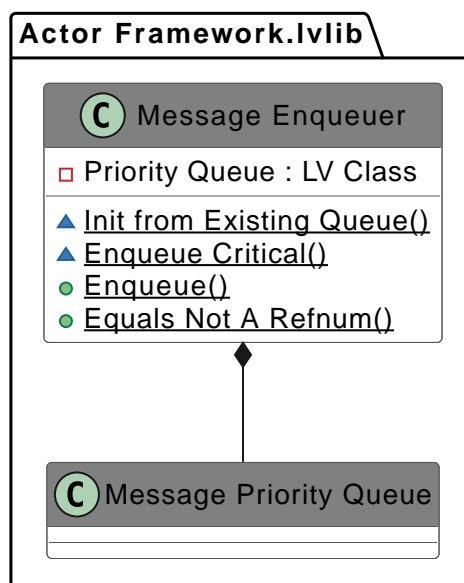
NOTE | No Constant VIs Found

3.10. Message Enqueuer.lvclass

Responsibility: No description found (add content in lvclass description)

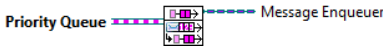







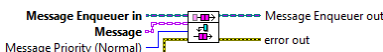





Version: 1.0.0.0

3.10.1. Diagram



3.10.2. Methods

Table 12. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Init from Existing Queue		No description found (add content in vi description)			
Enqueue Critical		No description found (add content in vi description)			
Enqueue		<p><p>Filename: Actor Framework.lvlib:Message Enqueuer.lvclass:Enqueue.vi</p></p> <p><p>Sends a message.</p></p>			
Equals Not A Refnum		<p><p>Filename: Actor Framework.lvlib:Message Enqueuer.lvclass:Equals Not A Refnum.vi</p></p> <p><p>Checks whether a message enqueuer reference is equal to Not a Refnum. Unlike the Not a Number/Path/Refnum? function, this function does not check whether a non-zero reference still is valid. Refer to the detailed help for more information.</p></p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.10.3. Class Constant VIs

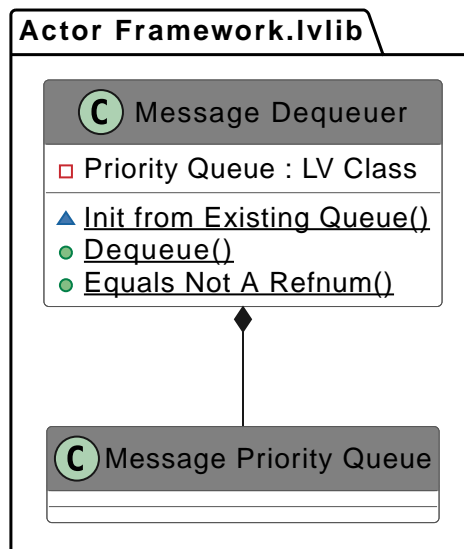
NOTE | No Constant VIs Found

3.11. Message Dequeuer.lvclass

Responsibility: No description found (add content in lvclass description)

Version: 1.0.0.0

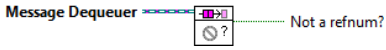


3.11.1. Diagram



3.11.2. Methods

Table 13. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Init from Existing Queue		No description found (add content in vi description)			
Dequeue		<p><p>Filename: Actor Framework.lvlib:Message Dequeueer.lvclass:Dequeue.vi</p></p> <p><p>Reads a message that a top-level actor sent to its caller. Use this VI to interact with non-actor code, not in a VI that belongs to an actor.</p></p>			

Name	Connector pane	Description	S.	R.	I.
Equals Not A Refnum		<p><p>(Filename: Actor Framework.lvlib:Message Dequeueer.lvclass:Equals Not A Refnum.vi)</p></p> <p><p>Checks whether a message dequeuer reference is equal to Not a Refnum. Unlike the Not a Number/Path/Refnum? function, this function does not check whether a non-zero reference still is valid. Refer to the detailed help for more information.</p></p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.11.3. Class Constant VIs

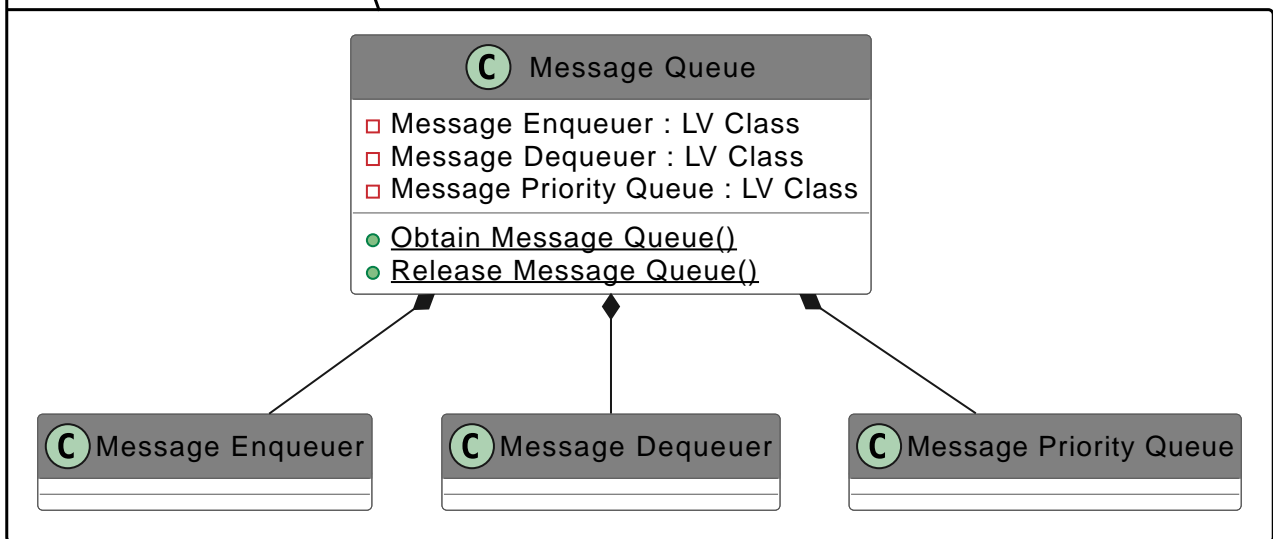
NOTE | No Constant VIs Found

3.12. Message Queue.lvclass

Responsibility: No description found (add content in lvclass description)

Version: 1.0.0.2

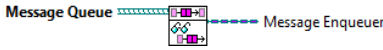



3.12.1. Diagram



3.12.2. Methods

Table 14. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Obtain Message Queue		<p><p>(Filename: Actor Framework.lvlib:Message Queue.lvclass:Obtain Message Queue.vi</p></p> <p><p>Obtains the reference to the message queue that the caller uses to communicate with the top-level actor.</p> </p>			
Read Dequeueer		<p><p>(Filename: Actor Framework.lvlib:Message Queue.lvclass:Read Dequeueer.vi</p></p> <p><p>Extracts the reference needed to read messages from the top-level actor.</p> </p>			

Name	Connector pane	Description	S.	R.	I.
Read Enqueuer		<p><p>(Filename: Actor Framework.lvlib:Message Queue.lvclass:Read Enqueuer.vi)</p></p> <p><p>Extracts the reference needed to send messages to the top-level actor.</p></p> <p><p>Use the Obtain Message Queue method to obtain the Message Queue input.</p></p>			
Release Message Queue		<p><p>(Filename: Actor Framework.lvlib:Message Queue.vi)</p></p> <p><p>Releases the reference to a message queue.</p></p> <p><p>Use the Obtain Message Queue method to obtain the Message Queue input.</p></p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.12.3. Class Constant VIs

NOTE No Constant VIs Found

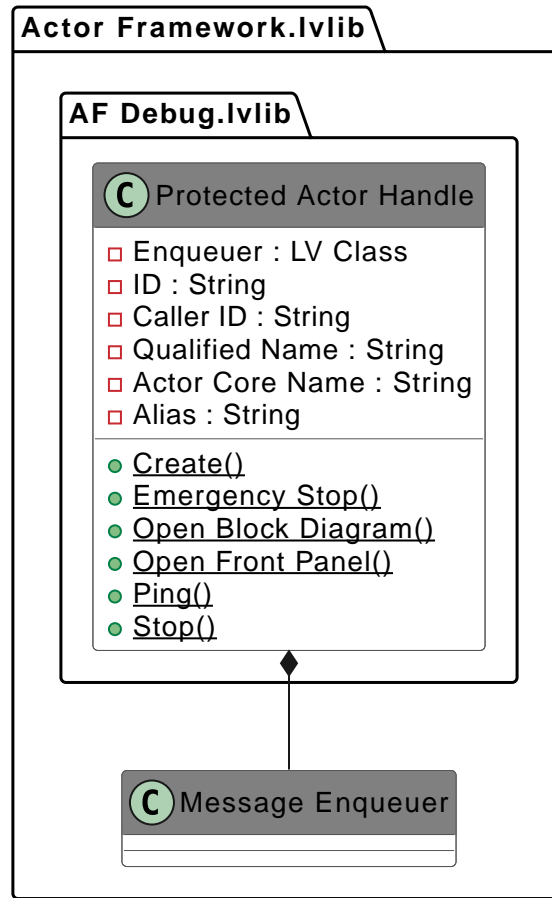
3.13. Protected Actor Handle.lvclass

Responsibility: An object of this class wraps access to an actor in order to limit a debugger's ability to interfere with the operation of the actors and to protect the debugger from actors stopping

unexpectedly.

Version: 1.0.0.0

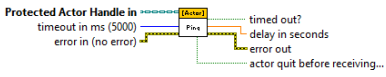



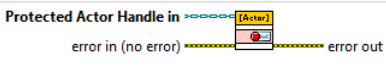
3.13.1. Diagram



3.13.2. Methods

Table 15. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Create	<p>This Actor [Actor] Protected Actor Handle out error in (no error) error out</p>	Given the information about an actor, provided by the debug layer of the Actor Framework, this VI creates an actor handle that an attached debugger can use for limited manipulation of the running actor.			
Emergency Stop	<p>Protected Actor Handle in [Actor] error in (no error) error out</p>	Sends the Stop Msg to the actor with emergency priority.			
Open Block Diagram	<p>Protected Actor Handle in [Actor] error in (no error) error out</p>	Opens the block diagram of the running actor's Actor Core clone VI.			
Open Front Panel	<p>Protected Actor Handle in [Actor] activate? (T) state (Standard) error in (no error) error out</p>	Opens the front panel of the running actor's Actor Core clone VI.			

Name	Connector pane	Description	S.	R.	I.
Ping		Sends the Ping Msg to the actor and waits for a reply to know how long the ping took to arrive. Use this VI in a debugger to test whether the actor is still alive.			
Read Details		Reads identification information about the actor, useful for display in a debugger.			
Stop		Sends the Stop Msg to the actor with normal priority.			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.13.3. Class Constant VIs

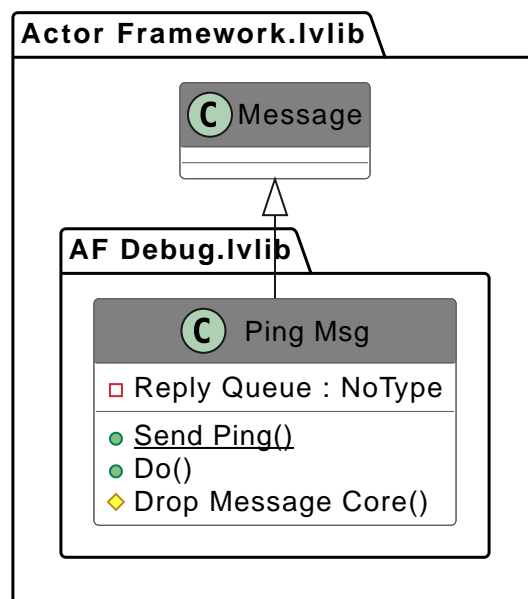
NOTE No Constant VIs Found

3.14. Ping Msg.lvclass

Responsibility: This message can be sent to any actor. It is a synchronous message: the Send method will not return until the message is received by the actor. When the message is received, the time between send and receive is computed using the high-resolution timer.

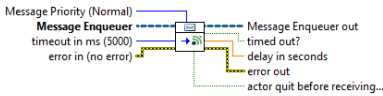




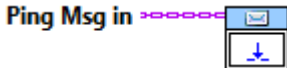


Version: 1.0.0.0

3.14.1. Diagram



3.14.2. Methods

Table 16. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Ping		This VI sends a synchronous message to any actor. This VI outputs the time delay between sending the message and the actor receiving the message (computed using the high-resolution timer). Delay will be zero if there is an error. Errors occur only if the actor has already quit before the ping is sent or if LabVIEW runs out of memory.			
Do		This VI delivers the Ping Msg to any actor. This VI does not invoke any methods on the actor. It simply returns to the sender the time delay (using the high-resolution timer) between the send and the receive of the message.			
Drop Message Core		<p>(Filename: Actor Framework.lvlib:Message.lvclass:Drop Message Core.vi)</p> <p>Defines what a message does if it is in the message queue when the queue is released. At this point, the actor has shut down, so it will never process the message.</p> <p>By default, this method does nothing. A child class may override it to define behavior. National Instruments provides the Reply Msg class to override this method. This class releases its internal queue, which causes any caller of the Send Message and Wait For Response method to stop waiting and return an error.</p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

3.14.3. Class Constant VIs

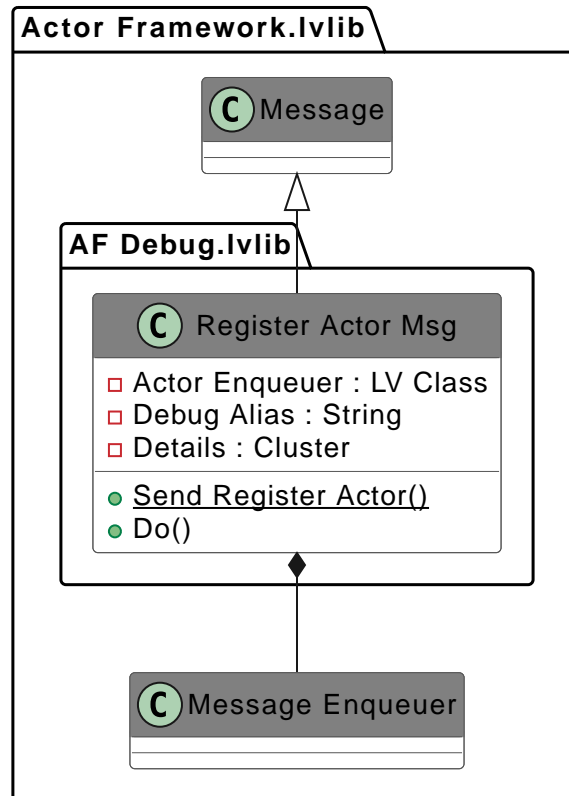
NOTE No Constant VIs Found

3.15. Register Actor Msg.lvclass

Responsibility: This message is sent to an actor as it is launching, before Pre-Launch Init even has a chance to execute. It will cause the Actor Core to register its clone number with the debugging tools, which will allow the debugger to perform operations such as opening the actor's block diagram.

Version: 1.0.0.0

3.15.1. Diagram



3.15.2. Methods

Table 17. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Send Register Actor	<p>Message Enqueuer (input), Message Enqueuer out (output), Debug Alias (input), Details (input), error in (no error) (input), error out (output)</p>	This VI sends the message to any actor to tell that actor to register its Actor Core clone name with the debugging tools.			
Do	<p>Register Actor (input), Actor in (input), Actor out (output), error in (no error) (input), error out (output)</p>	This VI delivers the Registration message to the actor. Unlike most messages, it does NOT invoke a method on the actor class.			

Scope: → Protected | → Community

Reentrancy: → Preallocated reentrancy | → Shared reentrancy

Inlining: → Inlined

3.15.3. Class Constant VIs

NOTE	No Constant VIs Found
-------------	-----------------------

Chapter 4. Actors (AF)

This section describes AF framework usage in the project

4.1. Preamble

Add anything that could be interesting to describe AF concepts and help the reader to understand the AF section

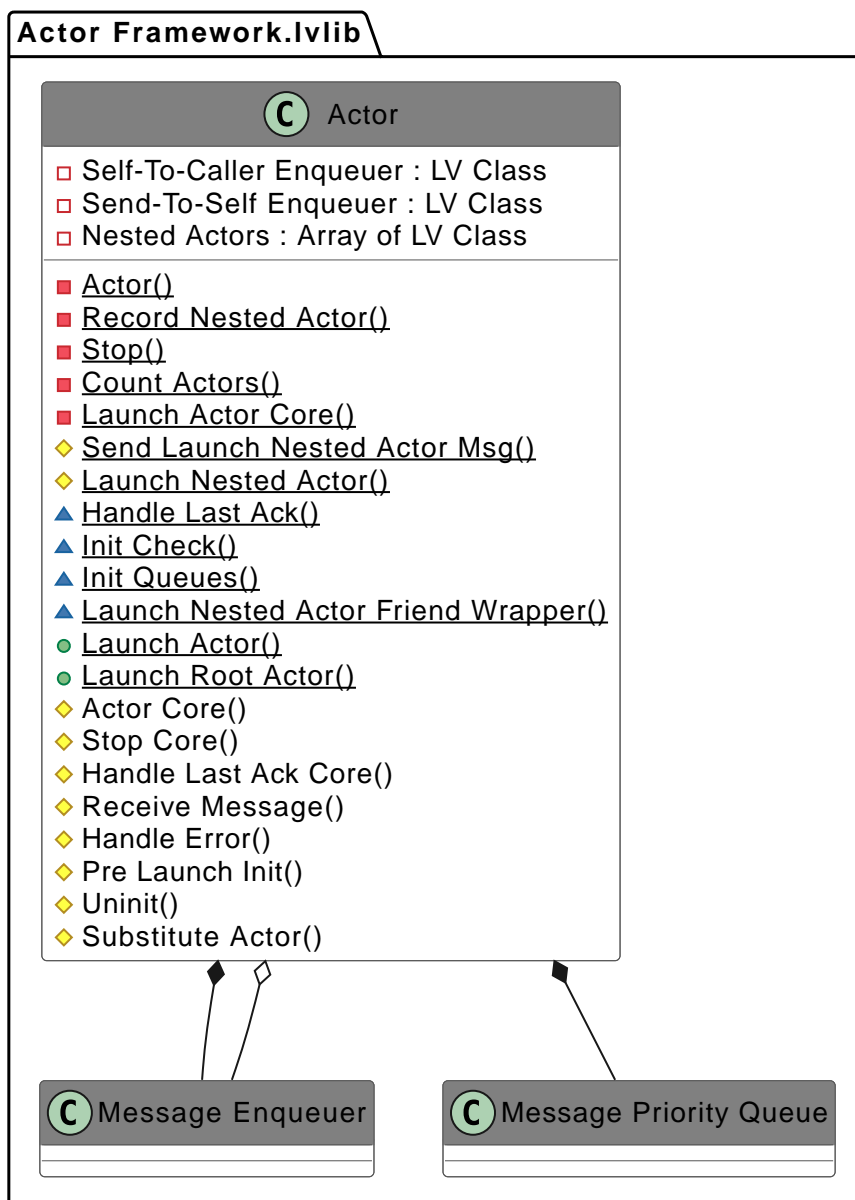
4.2. Actors overview

4.3. Actor.lvclass

Responsibility: This class serves as the base class for an independently running actor. The caller VI launches the actor using Launch Actor.vi and thereafter communicates with the actor using a pair of queues — one for messages to the actor, the other for messages from the actor. These messages are all derived from the Message.lvclass. Actor.lvclass provides mechanism for launching and establishing communications. In general, a programmer will create new classes that inherit from Actor.lvclass that are dedicated to specific purposes.

Version: 1.0.0.5















4.3.1. Diagram



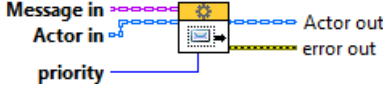











4.3.2. Methods




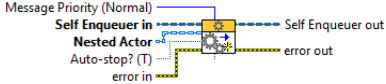

Table 18. Functions (non private scope only)

Name	Connector pane	Description	S.	R.	I.
Actor Core		<p>(Filename: Actor Framework.lvlib:Actor.lvclass:Actor Core.vi)</p> <p>Receives messages to the actor, reacts to them, and initiates any error handling that may shut the actor down.</p> <p>Descendant classes may override this VI to append parallel tasks for the actor to complete while it handles messages. Overrides use the Call Parent Class Method node.</p>			

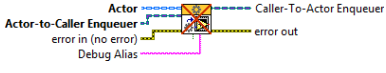

Name	Connector pane	Description	S.	R.	I.
Stop Core		<p>(Filename: Actor Framework.lvlib:Actor.lvclass:Stop Core.vi)</p> <p>Defines what the actor does before it stops. Use the final error code input to determine whether the actor shut down in response to an error.</p> <p>By default, this method passes the Stop message along to all auto-stop nested actors. A descendant class may override this method to define additional behavior, such as shutting down any processes the actor initiated in its override of the Actor Core method.</p>			
Handle Last Ack Core		<p>(Filename: Actor Framework.lvlib:Actor.lvclass:Handle Last Ack Core.vi)</p> <p>Defines how the caller actor responds to the Last Ack message from a nested actor. The Last Ack message is the final message a nested actor sends to its caller before it shuts down. The message contains information about the final state of the nested actor.</p> <p>By default, this method does nothing except return any error the nested actor sent. A descendant class may override this method to define behavior.</p>			
Read Self Enqueuer		<p><p>(Filename: Actor Framework.lvlib:Actor.lvclass:Read Self Enqueuer.vi)</p></p> <p><p>Returns the reference needed for the actor to send messages to itself.</p></p>			
Read Caller Enqueuer		<p><p>(Filename: Actor Framework.lvlib:Actor.lvclass:Read Caller Enqueuer.vi)</p></p> <p><p>Returns the reference the actor needs to send messages to its caller.</p></p>			

Name	Connector pane	Description	S.	R.	I.
Read Autostop Nested Actor Count		<p><p>(Filename: Actor Framework.lvlib:Actor.lvclass:Read Autostop Nested Actor Count.vi)</p></p> <p><p>Returns the number of auto-stop nested actors that were launched by an actor but have not yet sent back their Last Ack messages. Use this VI to detect whether all nested actors have finished working.</p></p>			
Receive Message		<p>This VI should be overridden ONLY if you are creating a proxy actor for remote transmission. If you are, override this VI to transmit the received message to a new destination, maintaining the given priority.</p>			
Handle Error		<p>(Filename: Actor Framework.lvlib:Actor.lvclass:Handle Error.vi)</p> <p>Defines how the actor handles an error that occurred while processing a message. By default, this method does nothing except stop the actor.</p>			
Pre Launch Init		<p>(Filename: Actor Framework.lvlib:Actor.lvclass:Pre Launch Init.vi)</p> <p>Defines behavior that occurs after the Launch Nested Actor method is invoked but before the actor's Actor Core method begins running.</p> <p>By default, this method does nothing. A descendant class may override it to define behavior.</p>			

Name	Connector pane	Description	S.	R.	I.
Uninit		<p>This method is the inverse of the Pre Launch Init.vi. This method is called unconditionally at the end of an actor's lifetime, even if there was an error in Pre Launch Init.vi. Note that the self Enqueuer for the actor is already invalid by the time this VI executes: within this method, the actor may send messages to its caller but not to itself. This method executes before Last Ack is sent.</p> <p>This method neither receives nor produces an error cluster. It is designed to provide an unconditional release of acquired resources. Any error raised during release is considered to be irrelevant to the actor's lifetime. If the actor truly needs to report some error status, it may send a message to caller or include information within itself that callers can examine when handling Last Ack.</p>			

Name	Connector pane	Description	S.	R.	I.
Substitute Actor		<p>(Filename: Actor Framework.lvlib:Actor.lvclass:Substitute Actor.vi)</p> <p>This method copies information from the Current Actor to the Substitute Actor, including the to-self and the to-caller queue references. The purpose of this function is to allow an actor to define a method that replaces itself with another actor, useful for implementing a State Pattern Actor. Child classes should override this method to copy any additional fields that they need from the current to the substitute.</p> <p>This method may return error code 678010: Illegal substitution. Overrides of this method may choose to return this error if the new actor is not one that is an appropriate substitute for the current actor, for whatever standards of "appropriate" are chosen for the current actor. If returning an error, override VIs should return the unmodified Current Actor as Substitute Actor out.</p>	 		
Send Launch Nested Actor Msg		<p><p>(Filename: Actor Framework.lvlib:Actor.lvclass:Send Launch Nested Actor Msg.vi)</p></p> <p><p>This VI sends a message containing an actor to another actor. The actor receiving the message will launch the payload actor as a nested actor. Use this VI only to send a message from an actor to itself.</p></p>			

Name	Connector pane	Description	S.	R.	I.
Launch Nested Actor		<p><p>(Filename: Actor Framework.lvlib:Actor.lvclass:Launch Nested Actor.vi)</p></p> <p><p>Launches an asynchronously running VI that performs tasks and handles messages for the Nested Actor. Use this VI to launch actors that are dependent on one or more calling actors. This VI returns a reference to the enqueueer that you can use to send messages to the newly launched actor.</p></p> <p><p>This VI requires the Caller Actor in input to call the Nested Actor. This VI will return an error if the Caller Actor in has not already been launched itself. Use the Launch Root Actor VI for launching an actor without a caller.</p></p>			
Handle Last Ack		<p>(Filename: Actor Framework.lvlib:Actor.lvclass:Handle Last Ack.vi)</p> <p>This VI is a wrapper around Handle Last Ack Core.vi. This wrapper is invoked by the Do method of the Last Ack class.</p>			
Init Check		No description found (add content in vi description)			
Init Queues		No description found (add content in vi description)			
Launch Nested Actor Friend Wrapper		<p>(Filename: Actor Framework.lvlib:Actor.lvclass:Launch Nested Actor Friend Wrapper.vi)</p> <p>This VI is a community-scoped wrapper around Actor Framework.lvlib:Actor.lvclass:Launch Nested Actor.vi. It is a behind-the-scenes component of the Actor Framework.</p>			

Name	Connector pane	Description	S.	R.	I.
Launch Actor		<p>LabVIEW no longer supports this VI. Use the Actor:Launch Root Actor VI or Actor:Launch Nested Actor VI instead.</p> <p>Launches a top-level VI that handles messages for an actor and starts this actor's tasks. This VI returns a reference to the queue you use to send messages to the actor.</p> <p>This VI requires the Actor-to-Caller Enqueuer input, which is the reference that allows the launched actor to communicate with the caller. If you are launching the top-level actor, use the Read Enqueuer method to obtain this reference. If you are launching a nested actor, use the Read Self Enqueuer method instead.</p>			
Launch Root Actor		<p><p>(Filename: Actor Framework.lvlib:Actor.lvclass:Launch Root Actor.vi</p></p> <p><p>Launches an asynchronously running VI that performs tasks and handles messages for the Actor</p>. This VI returns a reference to an enqueueer that you can use to send messages to the newly launched actor.</p></p> <p><p>The Launch Root Actor VI launches the Actor without a caller. Use this VI to launch the root actor of the actor tree. Use the Launch Nested Actor VI to launch all other actors.</p></p>			

Scope:  → Protected |  → Community

Reentrancy:  → Preallocated reentrancy |  → Shared reentrancy

Inlining:  → Inlined

4.3.3. Class Constant VIs

NOTE	No Constant VIs Found
-------------	-----------------------

Chapter 5. Legal Information

5.1. Document creation

This document has been generated using the following tools.

5.1.1. Antidoc

Project website: [Antidoc](#)

Maintainer website: [Wovalab](#)

BSD 3-Clause License

Copyright © 2019-2025, Wovalab, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

5.1.2. AsciiDoc for LabVIEW™

Project website: [AsciiDoc toolkit](#)

Maintainer website: [Wovalab](#)

BSD 3-Clause License

Copyright © 2019-2025, Wovalab, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

5.1.3. **classy Diagram Viewer**

Project website: [classy Diagram Viewer](#)

BSD 3-Clause License

Copyright © 2021, Tatiana Boyé All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

5.2. Product used in the project

Antidoc hasn't been able to detect third party products in the project. This is the author's responsibility to list any of the missing product used.