Passage du Cardinal 1  
1700 Fribourg  
Switzerland

Iterator Actor Documentation

|  |  |
| --- | --- |
| Document responsible: | Albert Adiyatullin |
| Last changed | 24.01.2018 |
| Description: | This document contains the documentation of the Iterator Actor framework. |

Table of Contents

[Description 2](#_Toc504576381)

[Methods 3](#_Toc504576382)

[Configure 3](#_Toc504576383)

[Stop Iterate 3](#_Toc504576384)

[Prepare 3](#_Toc504576385)

[Iterate 3](#_Toc504576386)

[Conclude 3](#_Toc504576387)

[Interaction with the framework 4](#_Toc504576388)

[Configure message 4](#_Toc504576389)

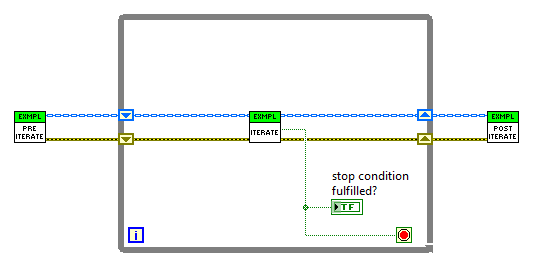
[Message for the Client Prepare method 4](#_Toc504576390)

[Error codes 4](#_Toc504576391)

# Description

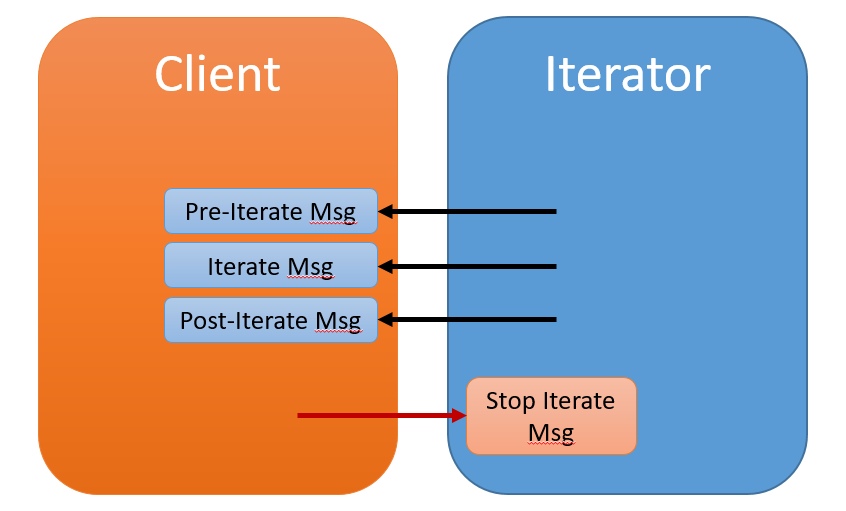
The Iterator Actor framework provides a possibility to stop the action loops in the Actor framework on demand.

If some actor method is called by a message and contains a loop like this:



then it cannot be stopped on demand, because any new message will be processed only after this method will complete.

The Iterator Actor is launched by the main (Client) actor and has messages to start the Pre-Iterate, Iterate, and Post-Iterate methods. Therefore, it can launch these methods, and at the same time be stopped by the Client actor.



The Iterator Actor is a state machine with four available states:

**Idle:** actor does not do anything.

**Preparation:** actor sends one Preparation message and waits for response. Depending on the response, it moves to another state.

**Iterating:** actor sends Iterate messages with a given delay between them. It moves to the Conclusion state either after a given number of iterations, or after the Client call.

**Conclusion:** actor sends one Conclusion message and moves to Idle state.

# Methods

All the methods are called using standard messages. Methods (and messages) Configure and Stop Iterate are public and available to the Caller; the other ones are only open to the community.

## Configure

Allows the user to give the handlers to the messages for preparation, iterating, and conclusion of the operations.

Changes the status to Preparation, calls the Prepare message.

## Stop Iterate

Allows the user to stop the iterations.

Changes the status to Conclusion, cancels the schedules Iterate messages (if present), calls the Conclude method.

## Prepare

Sends the Prepare message to the Caller and waits for a response. Depending on the response:

* No response: changes the status to Idle
* TRUE: changes the status to Iterating, calls the Iterate method
* FALSE: changes the status to Conclusion, calls the Conclude method

## Iterate

Sends the Iterate message to the Caller. Depending on the number of iterations left:

* -1: sends a time-delayed message for itself, thus scheduling a new Iteration
* 0: changes status to Conclusion, calls the Conclude method
* >0: sends a time-delayed message for itself, thus scheduling a new Iteration, decrements number of Iterations left

## Conclude

Sends the Conclude message to the Caller, changes the status to Idle, thus stopping the operation.

# Interaction with the framework

## Configure message

You need to provide:

* Three message classes that call your Preparation, Iteration, and Conclusion methods. All three classes should inherit from the Reply Msg (not from a standard Message), with only the method Do Core overridden. This is necessary for checking the timeouts of all the actions.
* Cluster with Iterator settings:
  + Timeouts for Preparation, each of Iteration, and Conclusion phase
  + Delay after the finished iteration to start a new one
  + Number of iterations (unlimited if -1, in this case it should be stopped by the Client.)

## Message for the Client Prepare method

This message should return a Boolean converted to a variant, indicating is it ready for the Iterations (TRUE) or not (FALSE).

# Error codes