Passage du Cardinal 1  
1700 Fribourg  
Switzerland

Cell Positioner Documentation

|  |  |
| --- | --- |
| Document responsible: |  |
| Last changed | 19.01.2018 |
| Description: | This document contains the documentation of the Event Actors framework. |

Table of Contents

[Description 2](#_Toc503974047)

[WebSocket 2](#_Toc503974048)

[WebSocket services 2](#_Toc503974049)

[Configuration file 3](#_Toc503974050)

[Instrument geometry 4](#_Toc503974051)

[Edge position 4](#_Toc503974052)

[Body position 4](#_Toc503974053)

[X/Y Stage Config File Section 4](#_Toc503974054)

[Cell Positioner API methods 4](#_Toc503974055)

[Connect Stages 4](#_Toc503974056)

[Disconnect Stages 4](#_Toc503974057)

[Initialize Position 4](#_Toc503974058)

[Go to Position 4](#_Toc503974059)

[Go to Body 4](#_Toc503974060)

[Go to Edge 5](#_Toc503974061)

[Exit 5](#_Toc503974062)

[Planned improvements 5](#_Toc503974063)

[Error codes 5](#_Toc503974064)

# Description

The Event Actor framework provides the way to send the messages between the actors at zero coupling between them.

This is done by using the event formalism, when the sender uses a generic message “Send event” without having the “Do” method. Therefore, the sender does not need to know any concrete implementations of the methods that the receiver will execute.

The interaction through events implies that for each event there is one concrete implementation of the method. Therefore the events are realized as classes, and the event handlers – as messages, containing only a “Handle Event” method (analogy of “Do” method in classic messages).

In the hierarchy of the framework, there is one Controller (master, previously called source) and many Views (slaves, previously called sinks). The whole list of Events that can be used for coupling are recorded at the Controller’s side. Each of them can be processed either by Controller or by View. The Views subscribe for the events that they would like to receive from the Controller.

# How to register an event

**New Sink event**

1. Create an event, an update method, and a message for it
   1. Create a class for the event type by inheriting from the Event class
   2. Create “Update …” method, where the control is updated using the reference from the Sink class private data and a property node.
   3. Create a message class for this method by inheriting from the “Event Generated Msg” class. Change the icon, otherwise you will not distinguish different messages during the event registration.
   4. Create a “Do” override method in the message class by copypasting any of the existing ones.
2. Let everyone know about the event
   1. Register this event in the “Register Events To Source” override method of the Sink class.
   2. Register this event in the “Define Events” override method in the Source (Controller) class.
   3. Do not forget to call the Generate Event method of the Event Source class in the Controller when you need to update the sink.

**New Source event**

1. Same (however, if you need no GUI update, just skip it).
2. Create an event, a method, and a message
   1. Create a class for the event type by inheriting from the Event class
   2. In the Source, create a method to process the event.
   3. Using the standard functionality (MGI Message Maker), create a message for this method. Then change the inheritance of the message to “Abstract Event Handler for Controller Actor”. The methods “Send …” and “Do” can be deleted – MGI Message Maker will not notice it. Change the icon, otherwise you will not distinguish different messages during the event registration.
   4. Create a “Handle Event” override method in the message class by copypasting any of the existing ones.
3. Let everyone know about the event.
   1. Register the event in the “Define Events” override method in the Controller class. Send True to the “Source Has Handler” input of the “Add Event” method, and provide the message class as a message handler.
   2. Create a method and a message for the View; the method should call “Generate Event” of the parent Event Sink class.