

ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-event



Peter Reutemann

December 24, 2014

©2012-2014



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato



Except where otherwise noted, this work is licensed under
<http://creativecommons.org/licenses/by-sa/3.0/>

Contents

1	Flow	7
2	Cron jobs	9
	Bibliography	11

List of Figures

2.1	The editor for entering the execution time of a cron job.	9
-----	---	---

Chapter 1

Flow

The following actors are available:

- *TriggerEvent* – control actor for triggering an event found below an *Events* standalone.
- *Cron* – standalone for scheduled events. See 2 for more details.
- *DelayedEvent* – standalone that executes its sub-flow after a predefined number of milli-seconds.¹
- *Events* – standalone similar to *CallableActors*, used for grouping events.
- *ExternalStandalone* – is a triggerable event and can be added to the *Events* standalone.
- *Flow* – a flow itself is a triggerable event as well.
- *LogEvent* – standalone that allows listening to global logging events and processing of the received log records.
- *QueueEvent* – standalone that allows listening to a queue (ArrayList) located in internal storage and process items as soon as they become available.
- *VariableChangedEvent* – standalone that allows listening to changes to a specific variable.

¹adams-event-delayed_event.flow

Chapter 2

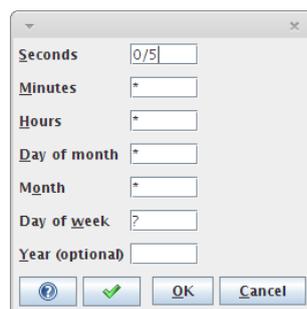
Cron jobs

Cron jobs are a very convenient way of performing background jobs that are only executed once in a while at specific times. For instance, a “clean up” job could run once every night, or a “backup” job every Friday night.

The *Cron* standalone can be used to *trigger* and execute these jobs. You can either place the actors that the job consists of below the *Cron* itself¹ or place them under a *Flow* control actor inside the *Events* standalone and call this event then using the *TriggerEvent* control actor.²

Since the cron jobs get executed at fixed times, we need to have a flow that is running forever, unless stopped by the user (or the cron job itself). This can be implemented using the *WhileLoop* control actor with an enclosed *Sleep actor*. To simplify this setup, there is already a sub-flow template available that generates such a flow: *EndlessLoop*.

The format for the cron job execution is not very intuitive when you look at it for the first time. In order to make it easier, you can use the editor that ADAMS offers (see Figure 2.1). This editor allows you to check the current input as well as opening a web page with a detailed description of the format (see [2]).



The image shows a dialog box for editing cron job execution times. It has a title bar with a close button (X) and a dropdown arrow. The fields are: Seconds (0/5), Minutes (+), Hours (+), Day of month (+), Month (+), Day of week (?), and Year (optional). At the bottom are four buttons: a help icon (?), a green checkmark, OK, and Cancel.

Figure 2.1: The editor for entering the execution time of a cron job.

¹adams-event-displaying_directory_contents1.flow

²adams-event-displaying_directory_contents2.flow

Bibliography

- [1] *ADAMS* – Advanced Data mining and Machine learning System
<https://adams.cms.waikato.ac.nz/>
- [2] *Cron format* – detailed explanation of format
<http://www.quartz-scheduler.org/docs/tutorials/crontrigger.html>