ADAMS

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

Module: adams-rats-core



Peter Reutemann

November 21, 2026

©2014-2019



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato



Except where otherwise noted, this work is licensed under
http://creativecommons.org/licenses/by-sa/4.0/

Contents

1 Introduction **5**

2 Flow **7**
2.1 Actors 7
2.2 Rat inputs 7
2.3 Rat outputs 8
2.4 Remote commands 8

Bibliography **11**

Chapter 1

Introduction

The *Reception And Transmission System*, or RATS for short, is aimed at scenarios where data is being received from various sources, processed and then transmitted to various destinations again. It simplifies the design of flows that handle these kind of scenarios, by providing off-the-shelf *receivers* and *transmitters*, e.g., for directory polling or FTPing files.

In contrast to regular ADAMS flows, the RATS sub-system is event-driven and not data-driven. The received data can then be processed by a data-driven flow before sending it somewhere else.

Chapter 2

Flow

2.1 Actors

The following standalone actors are available:

- *Rats* – This standalone encloses multiple RAT configurations.
- *Rat* – Definition of how to receive data and how to transmit it, based on the specified *RatInput* (= receiver) and *RatOutput* (= transmitter). See following sections.
- *RatControl* – displays a control panel in the Flow editor for all the Rat actors that are to be displayed (off by default).
- *RatPlague* – creates copies of itself, one for each of the defined input queues, feeding into the same output queue.
- *LabRat* – the actual Rat setup gets generated at runtime using a generator.

The following control actors are available:

- *ChangeRatState* – changes the state (eg RUNNING, PAUSED) of a *Rat* actor. This allows the startup of Rat actors in a paused state before activating them later on.
- *ExecuteRats* – allows the execution of Rat actor(s) in *MANUAL* mode whenever a token passes through.

2.2 Rat inputs

The following Rat inputs are available:

- *Cron* – executes the base input scheme on the specified schedule.
- *DeQueue* – pops off items from the specified queue as soon as they become available.
- *DirWatch* – triggers the base input when the specified directory events occur (eg file created or changed).
- *DummyCronInput* – used as a dummy for *Cron*.
- *DummyInput* – dummy, does nothing.
- *Exec* – executes an external command.
- *FileLister* – for generating a list of files in a directory.

- *InputPolling* – simple meta-input that waits for the specified period before executing the base input again.
- *InputWithCallableTransformer* – meta-input that passes the data through a callable transformer first before passing it on.
- *InputWithExternalTransformer* – meta-input that passes the data through the external transformer first before passing it on.
- *Storage* – retrieves the specified item from internal storage if available.
- *StringToken* – just forwards the specified string.
- *Subscribe* – subscribes to the specified pub/sub handler.
- *Variable* – outputs the value of the specified variable if available.

2.3 Rat outputs

The following Rat outputs are available:

- *BinaryFileWriter* – writes the data as binary blob to a file.
- *Branch* – fowards the same data to all of the defined outputs.
- *CallableActor* – forwards the data to the specified callable actor.
- *ContainerValuePicker* – extracts the specified value from the container and passes it on to the base output.
- *DistribuedEnQueue* – incoming data is distributed among specified output queues (simply iterating through queues).
- *DummyOutput* – dummy, does nothing with the data; can be used if there is no output generated.
- *EnQueue* – adds the incoming data to the specified queue.
- *Exec* – executes the command whenever data is received, but does nothing with the data.
- *FileMover* – moves the incoming files to the specified target directory.
- *OutputWithCallableTransformer* – meta-outout that passes the data through the callable transformer first before passing it on to base output.
- *OutputWithExternalTransformer* – meta-outout that passes the data through the external transformer first before passing it on to base output.
- *Publish* – publishes the data using the specified pub/sub handler.
- *QueueDistribute* – similar to DistributedEnQueue.
- *Serialize* – serializes the incoming data.
- *SimpleContainerContent* – extracts the content from the incoming simple container and forwards it to the base output.
- *Switch* – forwards the data to the sub-branch for which the corresponding condition evaluates to 'true'.
- *TextWriter* – just writes the incoming data to a text file.

2.4 Remote commands

The Rats module has some additional remote commands that allow the control of individual *Rat* actors, as long as they have been flagged to show up in a *RatControl* actor and such an actor is also present in the flow.

Available commands:

- *flow.GetRatControlStatus* – returns the status (stoppable/isstopped/pausable/ispaused) for all the registered Rat actors
- *flow.SendRatControlCommand* – sends the specified control command to a specific Rat actor (pause/resume/stop/start)

Bibliography

[1] *ADAMS* – Advanced Data mining and Machine learning System
`https://adams.cms.waikato.ac.nz/`