

---

# **taran Documentation**

***Release 0.0.1***

**Jon Hadfield <jon@lessknown.co.uk>**

August 14, 2016



<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>API</b>	<b>5</b>
2.1	taran.foreman . . . . .	5
2.2	taran.starter . . . . .	6
2.3	taran.worker . . . . .	7
	<b>Python Module Index</b>	<b>9</b>



< Description here >

Release v0.0.1.



---

## Installation

---

*Using pip package manager:*

```
$ pip install taran
```

*From source:*

```
$ git clone https://github.com/jonhadfield/taran
$ cd taran
$ python setup.py install
```





## 2.1 taran.foreman

---

*Foreman*([configuration]) A template for all decision processors.

---

The Foreman class - An abstraction of the AWS SWF Decider operations

**class** taran.foreman.**Decision** (*name, type, schedule\_to\_start\_timeout, start\_to\_close\_timeout, schedule\_to\_close\_timeout, task\_list, input*)

**\_\_getnewargs\_\_**()

Return self as a plain tuple. Used by copy and pickle.

**\_\_getstate\_\_**()

Exclude the OrderedDict from pickling

**\_\_repr\_\_**()

Return a nicely formatted representation string

**input**

Alias for field number 6

**name**

Alias for field number 0

**schedule\_to\_close\_timeout**

Alias for field number 4

**schedule\_to\_start\_timeout**

Alias for field number 2

**start\_to\_close\_timeout**

Alias for field number 3

**task\_list**

Alias for field number 5

**type**

Alias for field number 1

**class** taran.foreman.**Foreman** (*configuration=None*)

A template for all decision processors.

## configuration

*module*

The configuration a foreman needs in order to participate in the workflow.

**get\_activity\_results** (*activity=None*)

Get the result returned when the activity became completed.

**get\_workflow\_history** ()

Get entire workflow history.

**Returns** a dict containing the entire workflow execution history

**poll\_for\_decision\_task** ()

Poll for an decision task from SWF and return if a task token has been provided.

**Returns** **task** – Details of the assigned task.

**Return type** dict

**schedule\_activity\_tasks** (*decisions=None*)

Retrieve the workflow history.

**Args:** decisions (List): A list of dictionaries containing details of the activities to schedule.

**Parameters** **decisions** (list) –

## 2.2 taran.starter

---

*Starter*([configuration]) Class that defines instances of starter that are used to perform checks and then execute a workflow.

---

This module provides a class that abstracts the configuration and the SWF ‘start\_workflow\_execution’ operation.

**class** taran.starter.**Starter** (*configuration=None*)

Class that defines instances of starter that are used to perform checks and then execute a workflow.

**ensure\_activity\_type\_exists** (*activity\_name=None, activity\_version=None, activity\_task\_list=None*)

Check the activity type exists and create it if it doesn’t.

**Parameters**

- **activity\_name** (unicode) –
- **activity\_version** (unicode) –
- **activity\_task\_list** (unicode) –

**ensure\_domain\_exists** (*domain\_name=None*)

Return true if specified domain exists, otherwise create it.

**ensure\_workflow\_type\_exists** (*workflow\_name=None, workflow\_version=None*)

Check the workflow type exists and create it if it doesn’t.

**Parameters**

- **workflow\_name** (unicode) –
- **workflow\_version** (unicode) –

**start\_workflow** ()

Start the workflow.

## 2.3 taran.worker

---

*Worker*([configuration]) A template for all decision processors.

---

This module provides worker/actor specific methods to all child classes.

**class** `taran.worker.Worker` (*configuration=None*)

A template for all decision processors.

**configuration**

*module*

The configuration a worker needs in order to participate in the workflow.

**activity\_task\_failed** (*reason=None, details=None*)

Signal that activity task failed.

**complete\_activity\_task** (*result=u'Undefined'*)

Signal activity task as complete.

**get\_activity\_results** (*activity=None*)

Get a list of all results (when activity completed)

**poll\_for\_activity\_task** ()

Poll for an activity task from SWF and return if a task token has been provided.

**Returns** `task` – Details of the assigned task.

**Return type** `dict`



**t**

`taran.foreman`, 5  
`taran.starter`, 6  
`taran.worker`, 7



## Symbols

`__getnewargs__()` (taran.foreman.Decision method), 5  
`__getstate__()` (taran.foreman.Decision method), 5  
`__repr__()` (taran.foreman.Decision method), 5

## A

`activity_task_failed()` (taran.worker.Worker method), 7

## C

`complete_activity_task()` (taran.worker.Worker method), 7  
`configuration` (taran.foreman.Foreman attribute), 5  
`configuration` (taran.worker.Worker attribute), 7

## D

`Decision` (class in taran.foreman), 5

## E

`ensure_activity_type_exists()` (taran.starter.Starter method), 6  
`ensure_domain_exists()` (taran.starter.Starter method), 6  
`ensure_workflow_type_exists()` (taran.starter.Starter method), 6

## F

`Foreman` (class in taran.foreman), 5

## G

`get_activity_results()` (taran.foreman.Foreman method), 6  
`get_activity_results()` (taran.worker.Worker method), 7  
`get_workflow_history()` (taran.foreman.Foreman method), 6

## I

`input` (taran.foreman.Decision attribute), 5

## N

`name` (taran.foreman.Decision attribute), 5

## P

`poll_for_activity_task()` (taran.worker.Worker method), 7  
`poll_for_decision_task()` (taran.foreman.Foreman method), 6

## S

`schedule_activity_tasks()` (taran.foreman.Foreman method), 6  
`schedule_to_close_timeout` (taran.foreman.Decision attribute), 5  
`schedule_to_start_timeout` (taran.foreman.Decision attribute), 5  
`start_to_close_timeout` (taran.foreman.Decision attribute), 5  
`start_workflow()` (taran.starter.Starter method), 6  
`Starter` (class in taran.starter), 6

## T

`taran.foreman` (module), 5  
`taran.starter` (module), 6  
`taran.worker` (module), 7  
`task_list` (taran.foreman.Decision attribute), 5  
`type` (taran.foreman.Decision attribute), 5

## W

`Worker` (class in taran.worker), 7