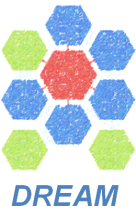


Developing with DREAM Platform

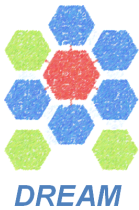
Nexedi 2015-03-26

Agenda



1. Install DREAM Runner
2. Edit DREAM platform source code
3. Run DREAM platform unit tests
4. Accessing DREAM GUI
5. Introduction to JSON format
6. Configuring Graph Editor
7. Configuring Application configuration, adding a spreadsheet tab and a new gadget
8. Configuring Knowledge extraction plugins
9. Cloud Execution of ACO

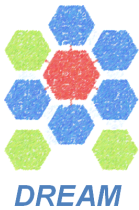
Install DREAM Runner



Easy way: use vifib hosting

1. Create an account on <https://www.slapos.org/>

Install DREAM Runner



Easy way: use vifib.com hosting

1. Create an account on <https://www.slapos.org/>
2. Request New Service

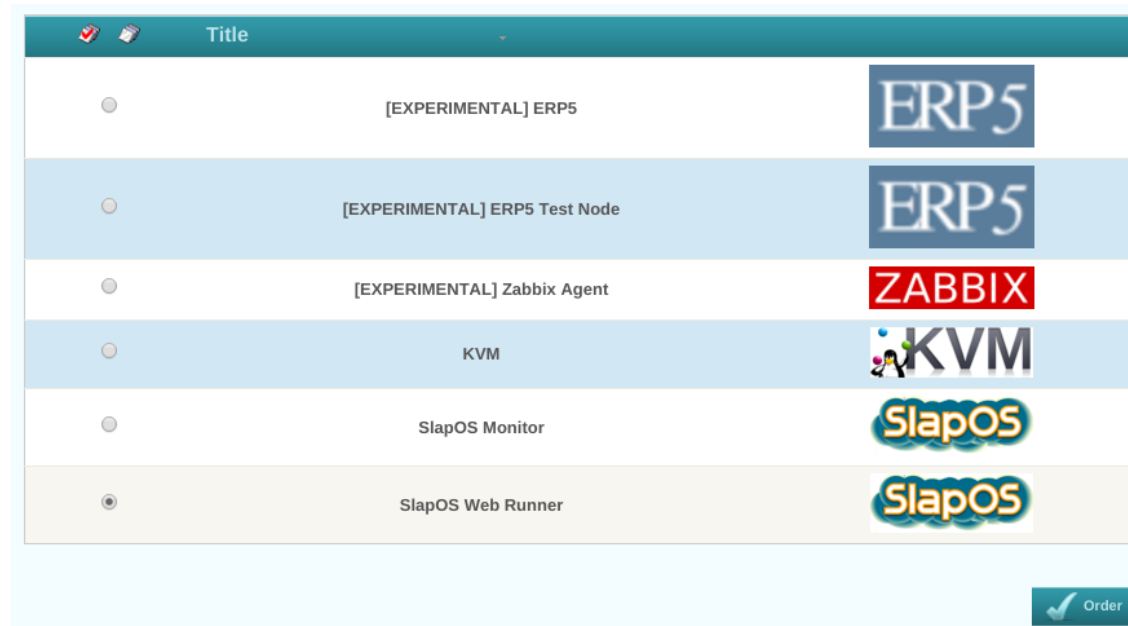


Install DREAM Runner



Easy way: use vifib.com hosting

1. Create an account on <https://www.slapos.org/>
2. Request New Service
3. Choose “SlapOS web runner”



Install DREAM Runner



Easy way: use vifib.com hosting

1. Create an account on <https://www.slapos.org/>
2. Request New Service
3. Choose “SlapOS web runner”
4. Choose latest version

<input type="radio"/>	SlapOS Web Runner (Outdated)	slapos-0.244	Production and Commercial version of SlapOS Web Runner
<input type="radio"/>	SlapOS Web Runner	slapos-0.254	Production and Commercial version of SlapOS Web Runner
<input type="radio"/>	SlapOS Web Runner (Outdated)	slapos-0.252.1	Production and Commercial version of SlapOS Web Runner
<input checked="" type="radio"/>	SlapOS Web Runner	slapos-0.258	Production and Commercial version of SlapOS Web Runner



Install DREAM Runner



Easy way: use vifib.com hosting

1. Create an account on <https://www.slapos.org/>
2. Request New Service
3. Choose “SlapOS web runner”
4. Choose latest version
5. Set Service Title and XML Parameter

SlapOS Web Runner

http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/tags/slapos-0.258:/software/slaprunner/software.cfg



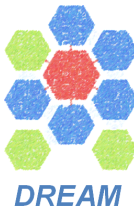
Service Title

DREAM PLATFORM

Parameter XML

```
<?xml version="1.0" encoding="utf-8"?>
<instance>
  <parameter id="slapos-software">software/dream</parameter>
  <parameter id="slapos-reference">dream</parameter>
  <parameter id="auto-deploy">true</parameter>
  <parameter id="user-authorized-key">(your ssh public key)</parameter>
  <parameter id="custom-frontend-backend-url">http://[${slaprunner:ipv6}]:8080</parameter>
</instance>
```


Install DREAM Runner



Easy way: use vifib.com hosting

1. Create an account on <https://www.slapos.org/>
2. Request New Service
3. Choose “SlapOS web runner”
4. Choose latest version
5. Set Service Title and XML Parameter
 - slapos-software: the software to use, available in slapos git repository, see <http://git.erp5.org/gitweb/slapos.git/tree/HEAD:/software?js=1>
 - slapos-reference: The git branch from slapos repository
 - auto-deploy: Auto deploy the software
 - user-authorized-keys: Your ssh public key, if you want to use ssh
 - custom-frontend-backend-url: This will generate an IPv4 frontend for you if you cannot access it through IPv6.

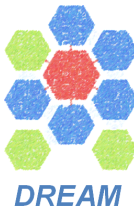
Install DREAM Runner



Easy way: use vifib.com hosting

1. Create an account on <https://www.slapos.org/>
2. Request New Service
3. Choose “SlapOS web runner”
4. Choose latest version
5. Set Service Title and XML Parameter
6. Wait for a while and connection parameter will appear

Install DREAM Runner



Easy way: use vifib.com hosting

1. Create an account on <https://www.slapos.org/>
2. Request New Service
3. Choose "SlapOS web runner"
4. Choose latest version
5. Set Service Title and XML Parameter
6. Wait for a while and connection parameter will appear
7. Click on **URL** parameter

Key	Value
backend_url	https://[2001:67c:1254:e:1c::b4c0]:50005
2_info	In order to set up your account, get the recovery-code from notification on monitor_info.
ssh_command	ssh 2001:67c:1254:e:1c::b4c0 -p 22222
url	https://softinst59074.host.vifib.net
public_url	https://softinst59072.host.vifib.net/public/
webdav_url	https://softinst59072.host.vifib.net/share/
git_public_url	https://[2001:67c:1254:e:1c::b4c0]:9684/git-public/
git_private_url	https://[2001:67c:1254:e:1c::b4c0]:9684/git/
custom-frontend-url	https://softinst59073.host.vifib.net
access_url	https://softinst59074.host.vifib.net/login
monitor_backend_url	https://[2001:67c:1254:e:1c::b4c0]:9684
1_info	On your first run, Use "access_url" to setup you account. Th "backend_url" if you want to use ipv6. Set up your account being able to clone your git repositories from the runner.
monitor_url	https://softinst59072.host.vifib.net

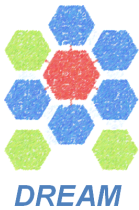
Install DREAM Runner



Easy way: use vifib.com hosting

1. Create an account on <https://www.slapos.org/>
2. Request New Service
3. Choose “SlapOS web runner”
4. Choose latest version
5. Set Service Title and XML Parameter
6. Wait for a while and connection parameter will appear
7. Click on URL parameter
8. ***Pay invoices every month***

Install DREAM Runner



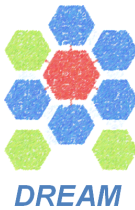
... or Install SlapOS in your IT infrastructure.

Follow the tutorials from:

<http://community.slapos.org/wiki/developer-Allocate.SlapOS.Master.Instance>

<http://community.slapos.org/wiki/developer-Installing.SlapOS.Package>

Install DREAM Runner



Before start, configure monitoring from “Monitoring” link on the slapos page.

1. Set monitoring password

This is the monitoring interface

Please set your password for later access

Password*:

Verify Password*:

Access

Install DREAM Runner



Before start, configure monitoring from “Monitoring” link on the slapos page.

1. Set monitoring password
2. Set recovery code

Monitoring

MONITOR-PUBLIC
rssfeed.html
MONITORING
status-history.cgi
status.cgi
ZERO-KNOWLEDGE
settings.cgi
monitor-password.cgi
cors-domain.cgi

Values that can be defined :

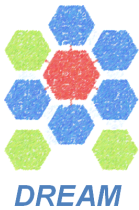
recovery-code	<input type="text" value="uamdntxf"/>
status-history-length	<input type="text" value="5"/>
shell-password	<input type="text" value="flcymqwh"/>



Save

Other values :

Install DREAM Runner



Upon first login, configure your account
Set your name, email, login and
password

Enter Password Recovery Code from
monitoring interface

A screenshot of the DREAM Runner web interface. The top navigation bar includes icons for home, a folder, and a user profile, followed by the text "Update your account - No account". Below this is a tabbed interface with "Editor", "Services", "Logs", "Terminal", and "Git". The "Editor" tab is active, showing a form titled "Your personal information". The form contains several input fields: "Your name:", "Your email address:", "User name:", "Password:", "Confirm Password:", and "Password Recovery code:". A "help ?" link is next to the "Password Recovery code:" field. At the bottom right of the form is a button labeled "Update Account".

Update your account - No account

Editor Services Logs Terminal Git

Your personal information

Your name:

Your email address:

User name:

Password:

Confirm Password:

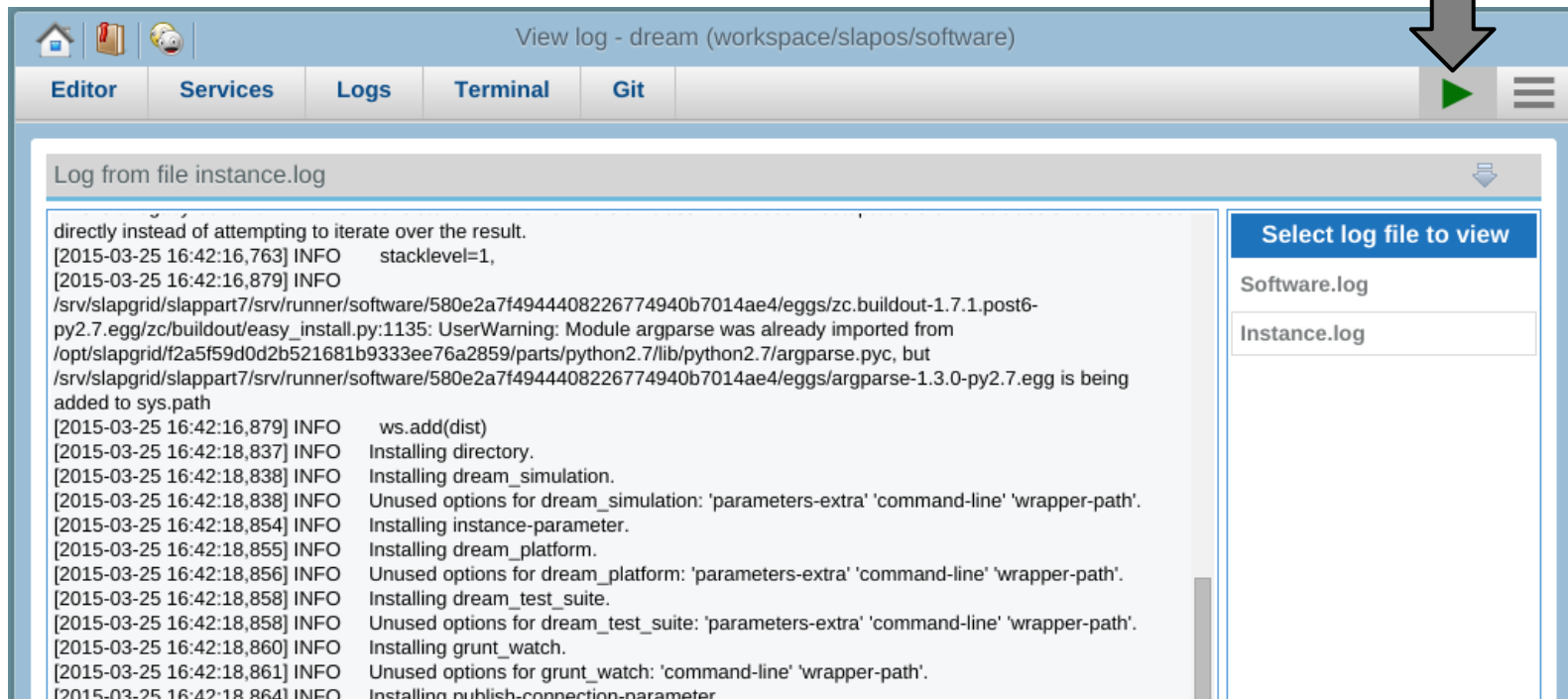
Password Recovery code: [help ?](#)

Update Account

Install DREAM Runner



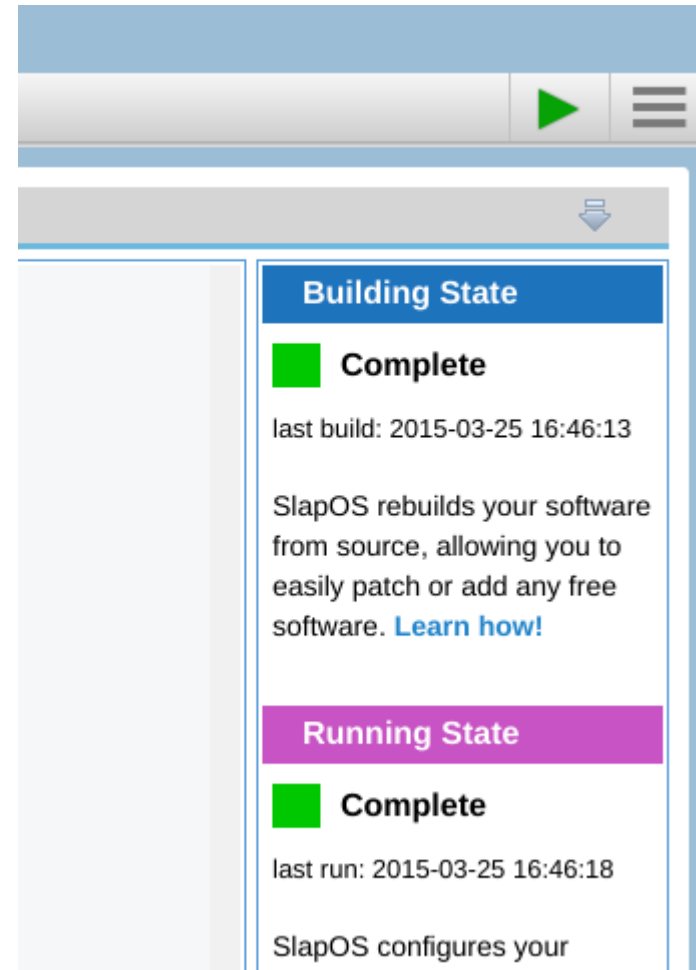
Click play button to compile software and create instance



Install DREAM Runner



Wait for building state and running state to be Complete



Install DREAM Runner



Check your processes are running in “Process” tab.
From this screen, you can also restart processes.

Instance inspection - dream (workspace/slapos/software)

Editor Services Logs Terminal Git

Process Connection Information Parameters Partitions Content Monitoring

This tab show all process generated by slapgrid for your application. You can click on the process name to display log.

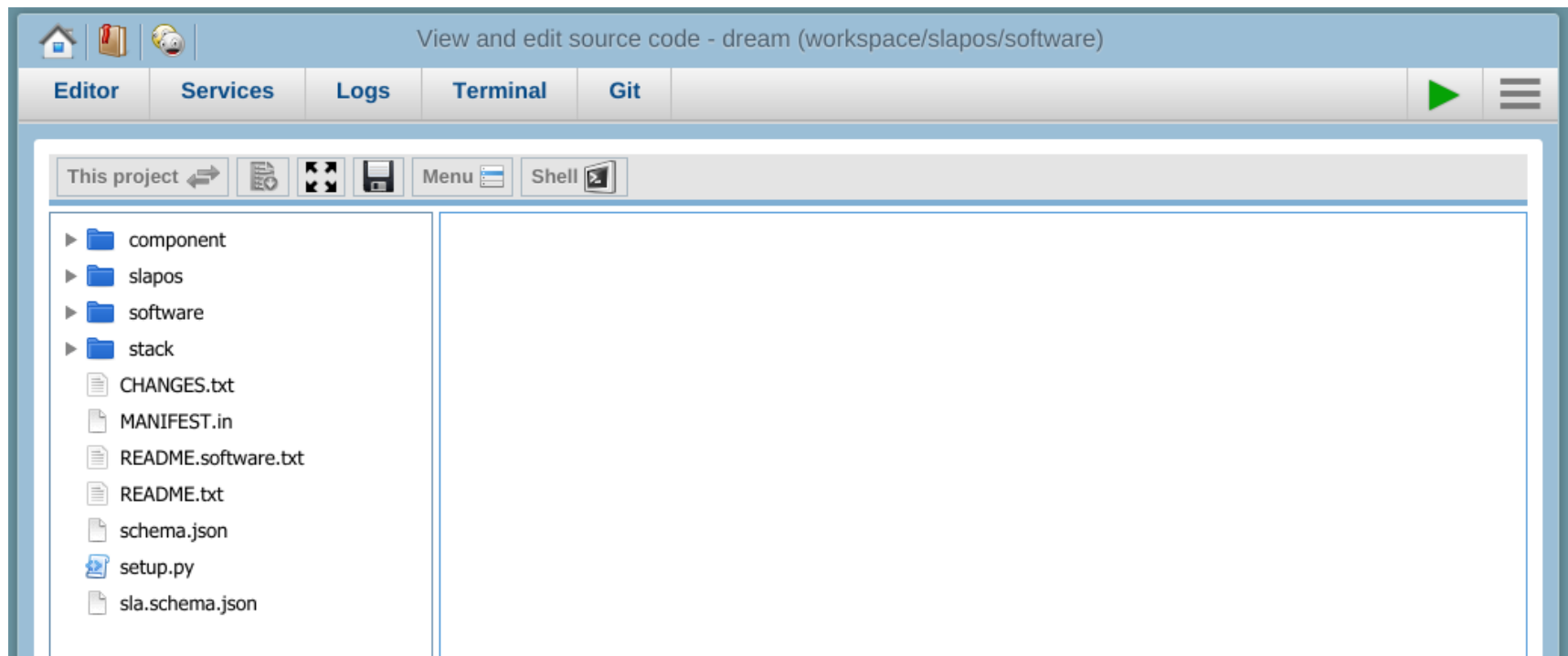
Partition and Process name	Status	Process PID	UpTime	
slappart0:dream_grunt_watch-on-watch	RUNNING	12567	0:04:52	Restart
slappart0:dream_platform-on-watch	RUNNING	12538	0:04:52	Restart
slappart0:dream_simulation	EXITED	25	PM	Restart
slappart0:dream_test_suite	RUNNING	18925	0:00:53	Restart

Refresh Status Stop all process

Edit DREAM platform source code



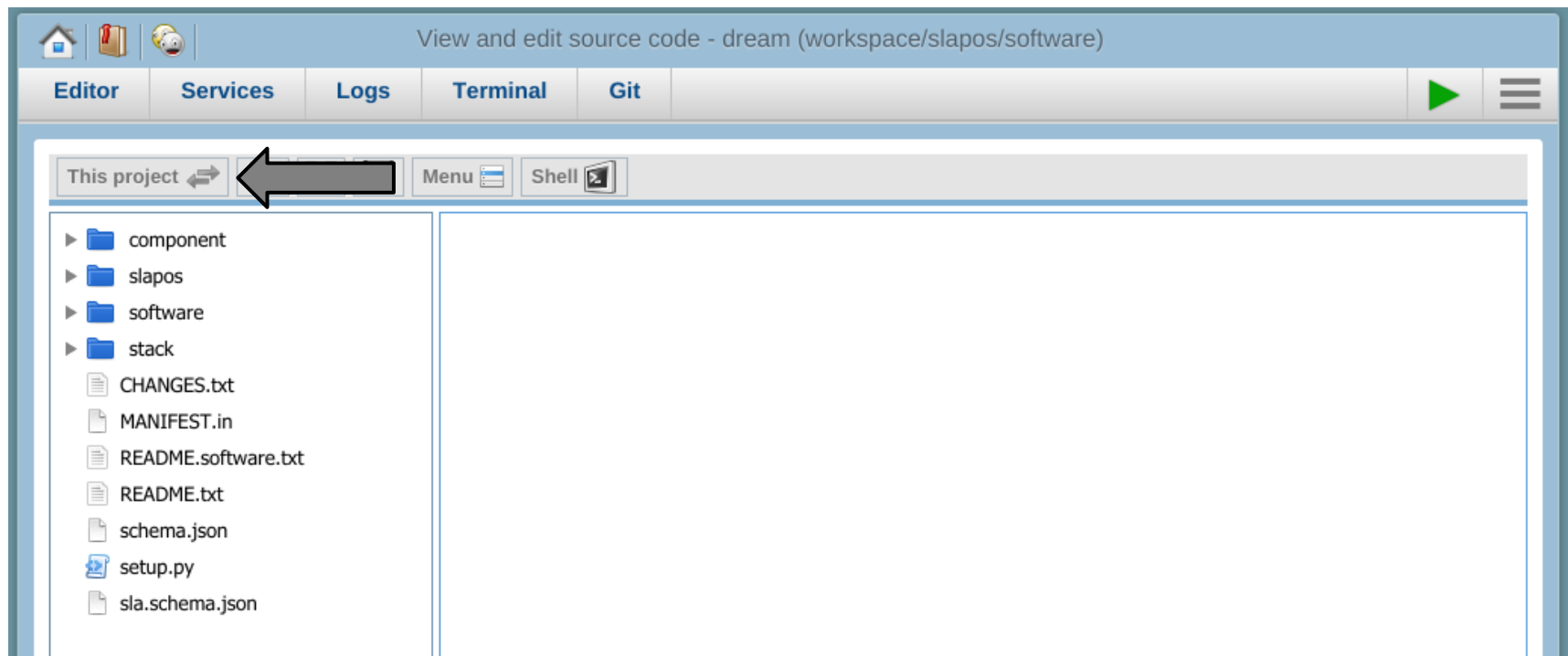
Editor tab let you edit files:



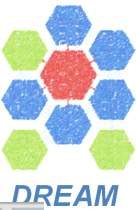
Edit DREAM platform source code



To access DREAM source code, first switch from “This project” to “Working dir”



Edit DREAM platform source code



DREAM source code is in:

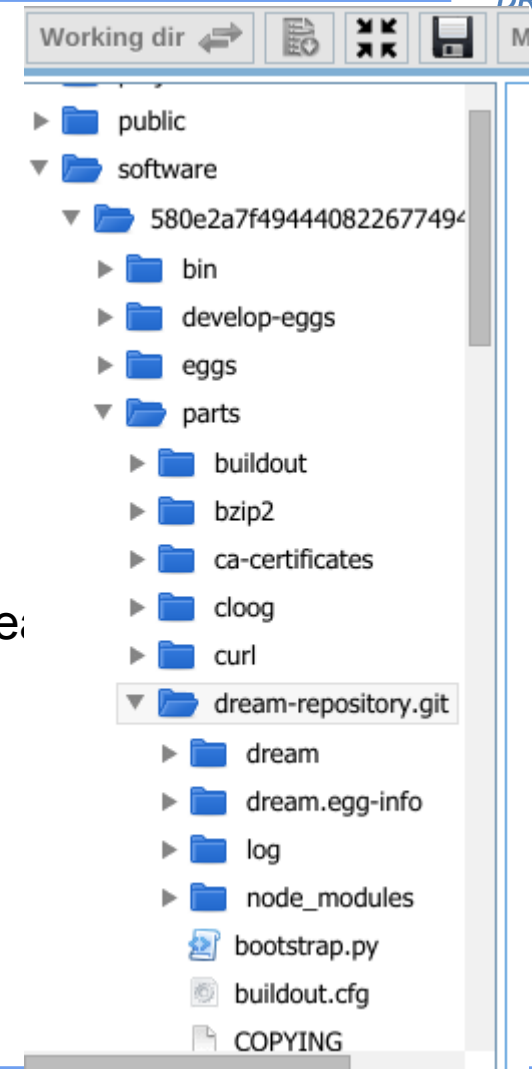
software

(the hash of the software)

parts

dream-repository.git

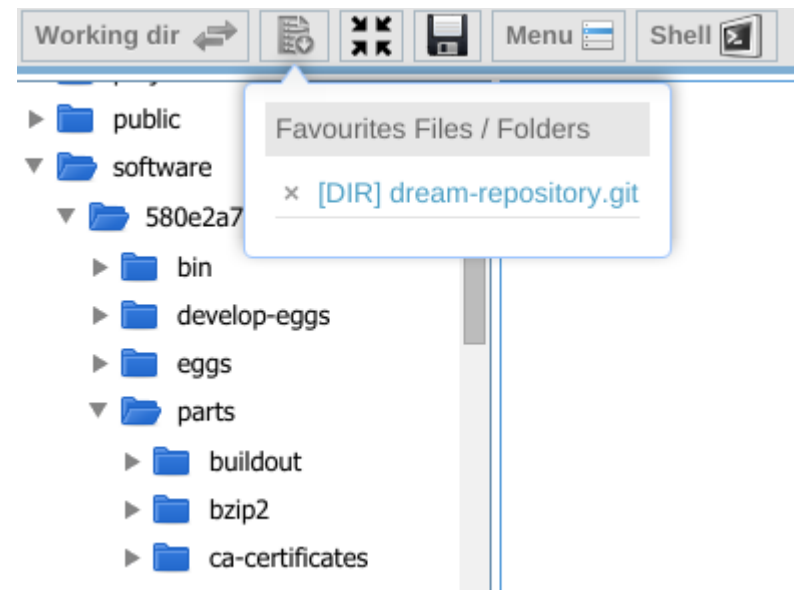
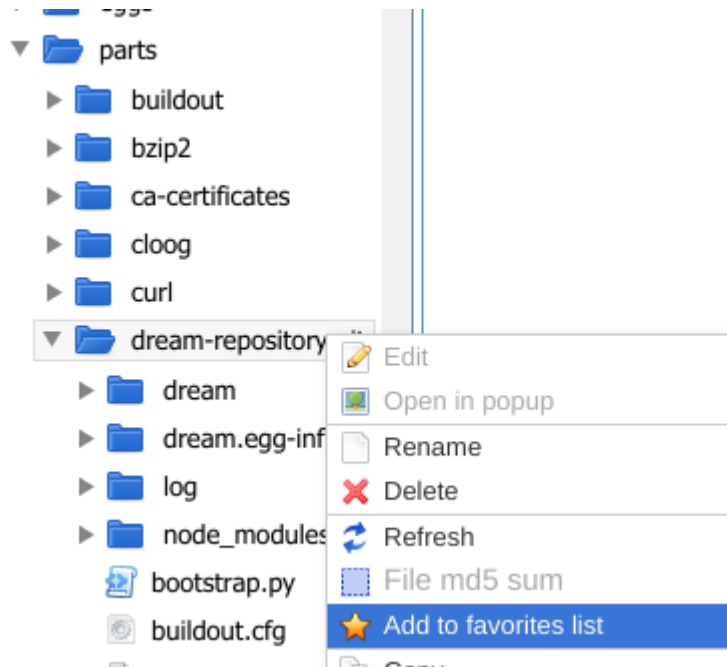
This is a git working copy of <http://git.erp5.org/gitweb/dream-repository.git/>



Edit DREAM platform source code



Tips: add this folder for favourites, it will be available from the favourites menu



Edit DREAM platform source code

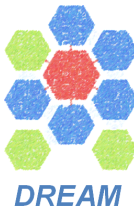


ManPy code can be edited :

```
Working dir [icon] [icon] [icon] [icon] Menu [icon] Shell [icon] Machine.py x
nndTopologies.py
Frame.py
FutureDemandCreator.py
Globals.py
imports.py
Job.py
JobMA.py
LineClearance.py
LineGenerationCMSD.py
LineGenerationJSON.py
M3.py
Machine.py
MachineJobShop.py
MachineManagedJob.py
ManPyObject.py
Mould.py
MouldAssembly.py

1 # =====
2 # Copyright 2013 University of Limerick
3 #
4 # This file is part of DREAM.
5 #
6 # DREAM is free software: you can redistribute it and/or modify
7 # it under the terms of the GNU Lesser General Public License as
8 # published by the Free Software Foundation, either version 3 of the
9 # License, or (at your option) any later version.
10 #
11 # DREAM is distributed in the hope that it will be useful,
12 # but WITHOUT ANY WARRANTY; without even the implied warranty of
13 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
14 # GNU Lesser General Public License for more details.
15 #
16 # You should have received a copy of the GNU Lesser General Public
17 # License along with DREAM. If not, see <http://www.gnu.org/licenses/>.
18 # =====
19 '''
20 Created on 8 Nov 2012
21
22 @author: George
23 '''
24
25 '''
26 Models a machine that can also have failures
27 '''
28
29 # from SimPy.Simulation import Process, Resource, SimEvent
30 # from SimPy.Simulation import activate, deactivate, waituntil
```


Running DREAM test suite



After making changes, run tests to validate your changes

Process

Connection Information

Parameters

Partitions Content


Monitoring

This tab show all process generated by slapgrid for your application. You can click on the process name to display log.

Partition and Process name	Status	Process PID	UpTime	
slappart0:dream_grunt_watch-on-watch	RUNNING	12567	0:23:24	Restart
slappart0:dream_platform-on-watch	RUNNING	12538	0:23:24	Restart
slappart0:dream_simulation	EXITED	25	PM	Restart
slappart0:dream_test_suite	EXITED	25		Restart

Refresh Status

Stop all process



Running DREAM test suite



After making changes, run tests to validate your changes

Process

Connection Information

Parameters

Partitions Content

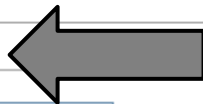
Monitoring

This tab show all process generated by slapgrid for your application. You can click on the process name to display log.

Partition and Process name	Status	Process PID	UpTime	
slappart0:dream_grunt_watch-on-watch	RUNNING	12567	0:23:24	Restart
slappart0:dream_platform-on-watch	RUNNING	12538	0:23:24	Restart
slappart0:dream_simulation	EXITED	25	PM	Restart
slappart0:dream_test_suite	EXITED	25	PM	Restart

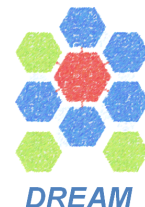
Refresh Status

Stop all process



2. Click on the process name to view the output

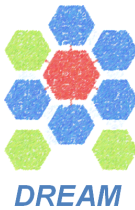
Accessing DREAM GUI



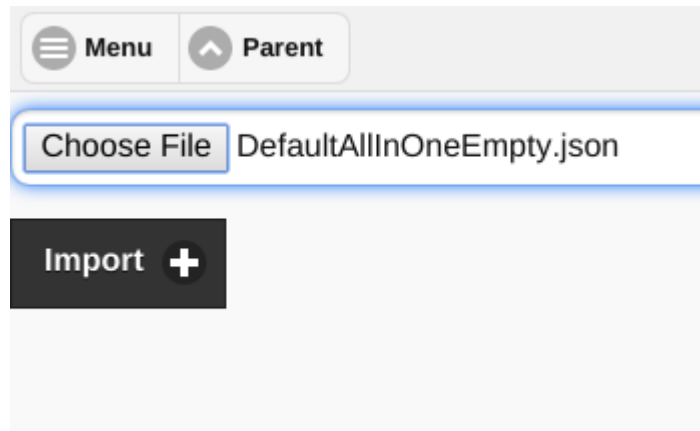
The GUI instance can be accessed from ***custom frontend URL*** in SlapOS parameter page.

Key	Value
backend_url	https://[2001:67c:1254:e:1c::b4c0]:50005
2_info	In order to set up your account, get the recovery-code from the monitoring interface. Before read the notification on monitor_info.
ssh_command	ssh 2001:67c:1254:e:1c::b4c0 -p 22222
url	https://softinst59074.host.vifib.net
public_url	https://softinst59072.host.vifib.net/public/
webdav_url	https://softinst59072.host.vifib.net/share/
git_public_url	https://[2001:67c:1254:e:1c::b4c0]:9684/git-public/
git_private_url	https://[2001:67c:1254:e:1c::b4c0]:9684/it/
custom-frontend-url	https://softinst59073.host.vifib.net
access_url	https://softinst59074.host.vifib.net/login
monitor_backend_url	https://[2001:67c:1254:e:1c::b4c0]:9684
1_info	On your first run, Use "access_url" to setup you account. Then you can use both "url" or "access_url". Or "backend_url" if you want to use ipv6. Set up your account in the webrunner in order to use webdav, and being able to clone your git repositories from the runner.
monitor_url	https://softinst59072.host.vifib.net

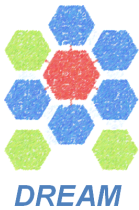
Accessing DREAM GUI



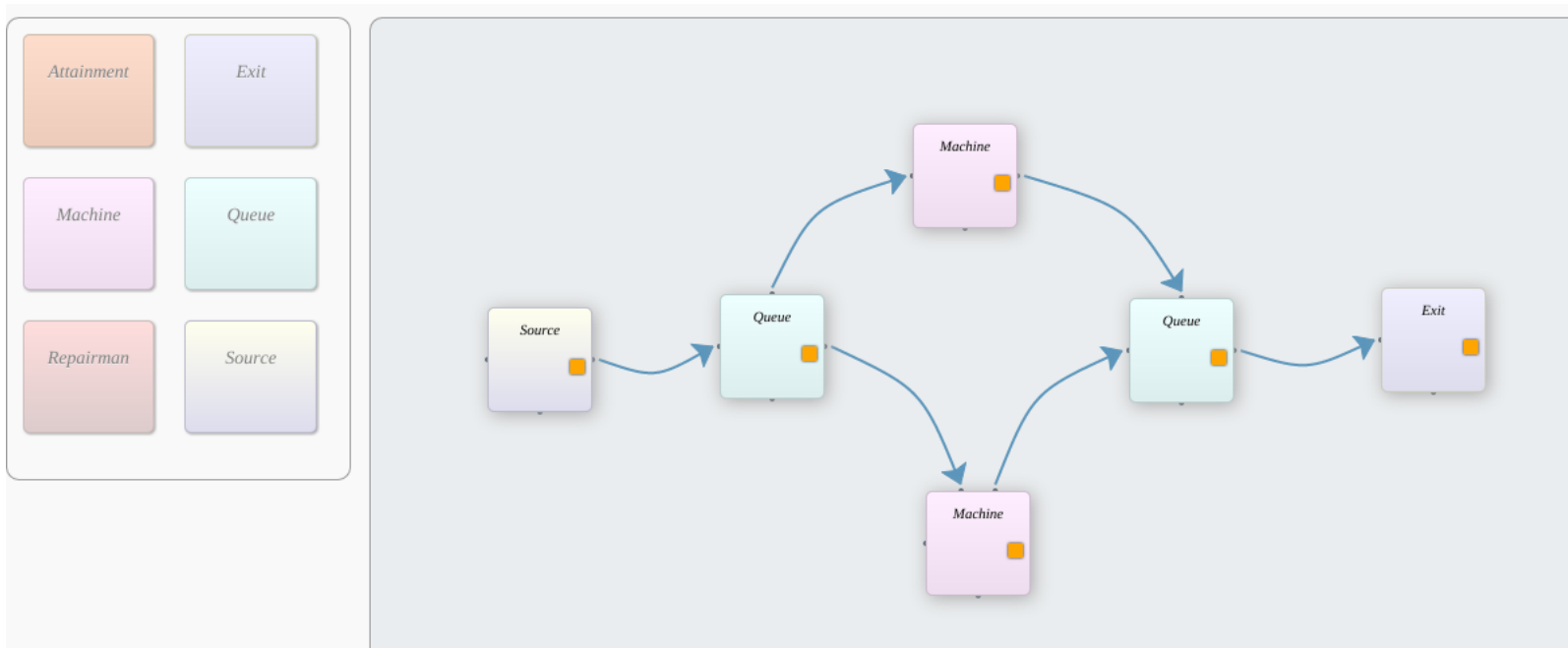
Load a model, for example https://raw.githubusercontent.com/nexedi/dream/master/dream/simulation/Examples/GUI_instances/DefaultAllInOneEmpty.json



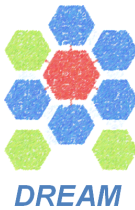
Accessing DREAM GUI



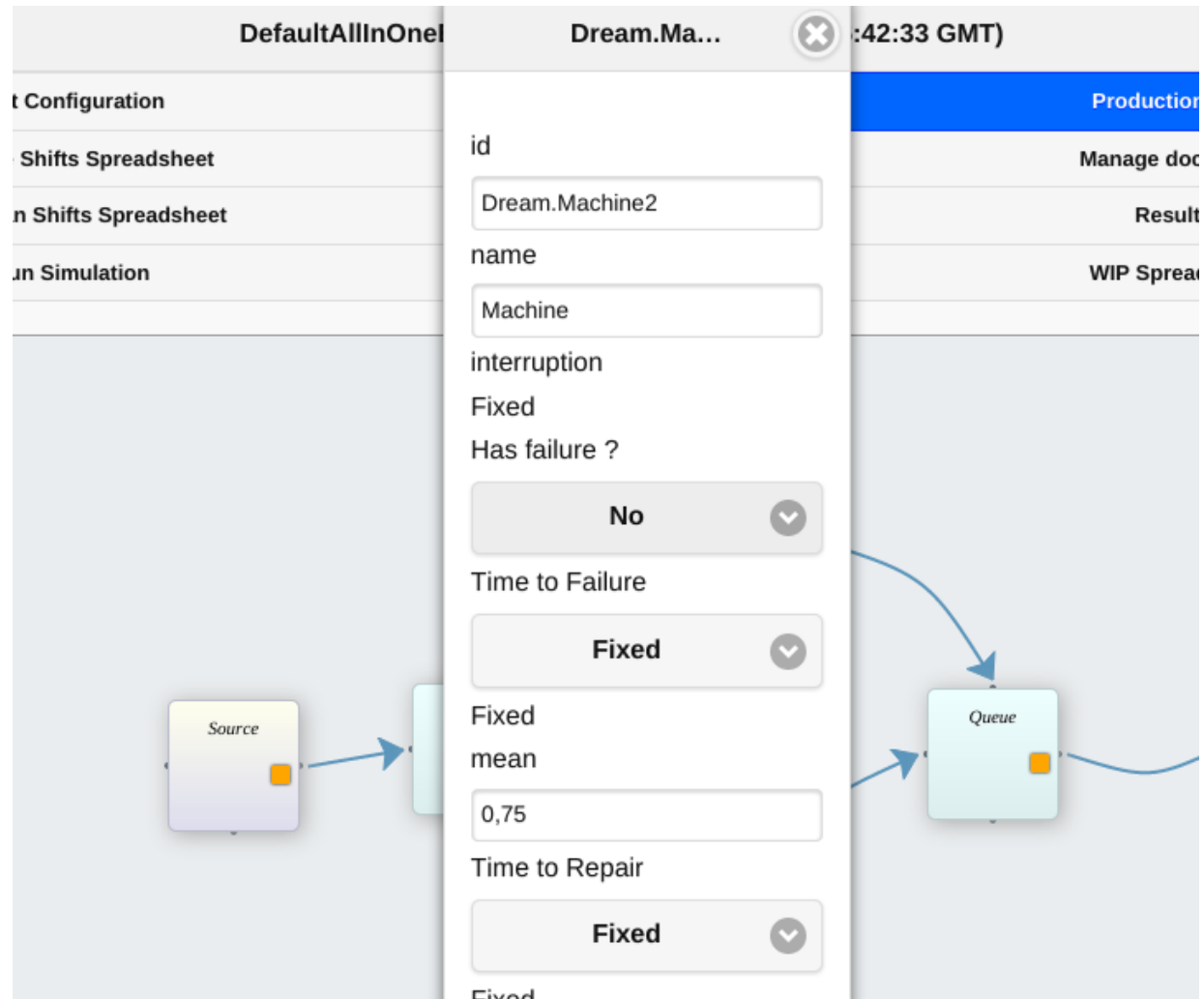
Create a simple model



Accessing DREAM GUI



Edit model properties



Accessing DREAM GUI



Running simulation

Machine Shifts Spreadsheet	
Repairman Shifts Spreadsheet	
Run Simulation	

Run Simulation ↻

Confidence level for statistical analysis of stochastic experiments

0,95

The day the experiment starts, in YYYY/MM/DD HH:MM:SS format

2015/01/01 09:00:00

The URL for knowledge extraction to access its data for example http://git.erp5.org/gitweb/dream.git/blob_plain/HEAD:/drear

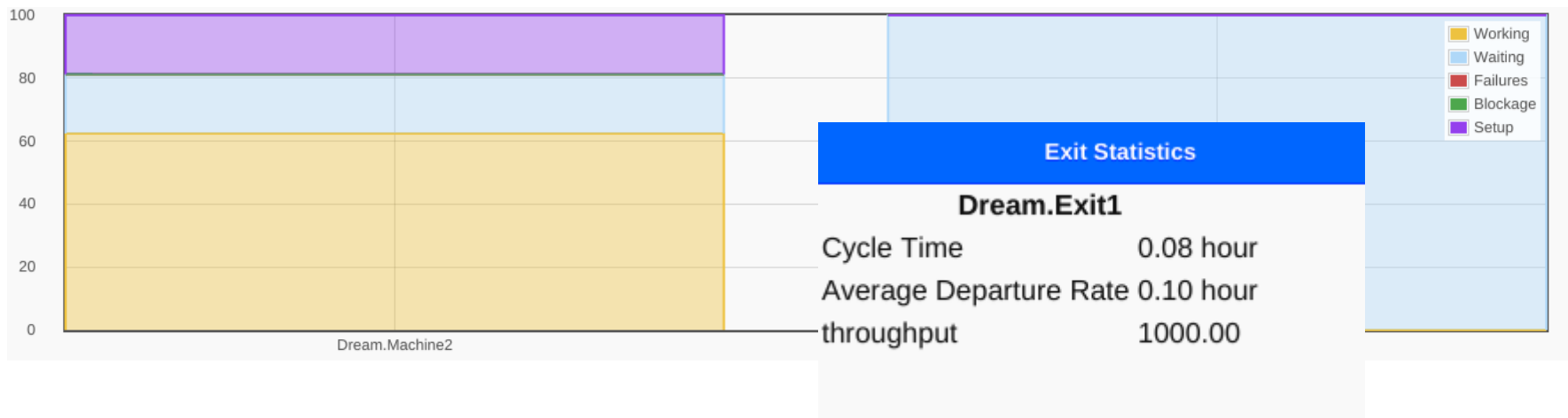
Length of the simulation run

100

Accessing DREAM GUI

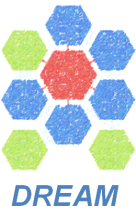


Visualise results



Exit Statistics		Queue Statistics	Station Utilization	Tabular_Results
Exit Id	Throughput	Takt Time	Lifespan	
Dream.Exit1	1000	0.09989999999999986	0.08125000000000017	

Introduction to JSON format

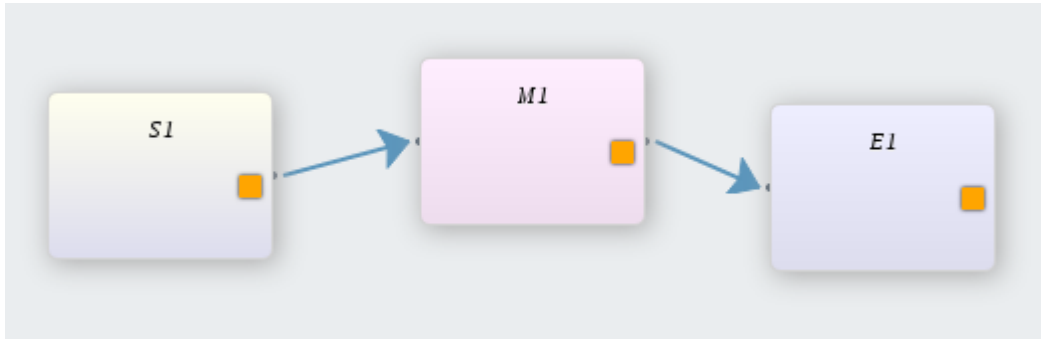


Only nodes

```
{
  "node": {
    "S1": {
      "name": "S1"
    },
    "M1": {
      "name": "M1"
    },
    "E1": {
      "name": "E1"
    }
  },
  "edge": { }
}
```

Simple Representation of a Directed Graph

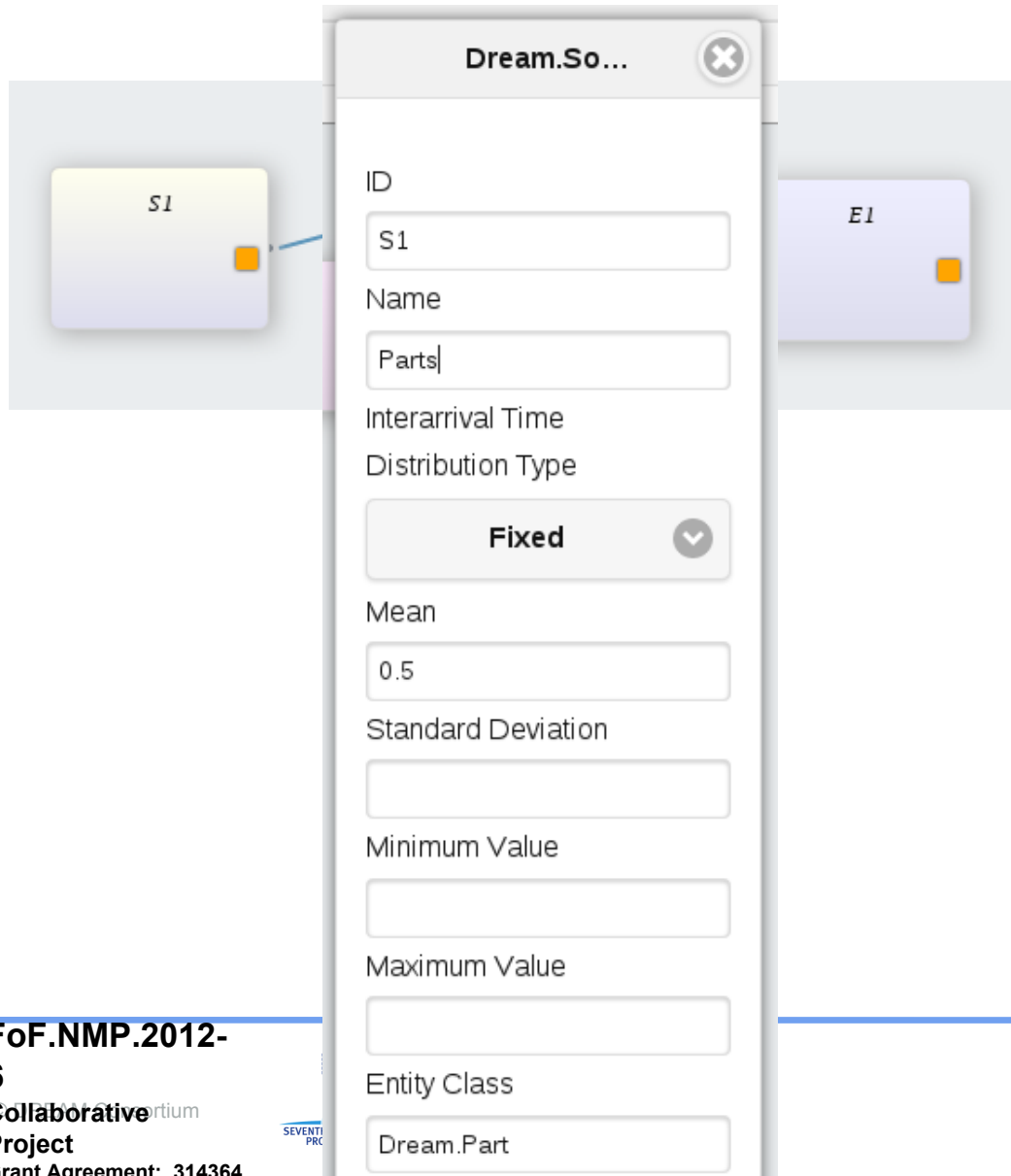
Introduction to JSON format



Nodes connected by edges

```
{
  "node": {
    "S1": {
      "name": "S1"
    },
    "M1": {
      "name": "M1"
    },
    "E1": {
      "name": "E1"
    }
  },
  "edge": {
    "edge1": {
      "source" : "S1",
      "destination" : "M1",
    },
    "edge2": {
      "source" : "M1",
      "destination" : "E1",
    }
  }
}
```


Introduction to JSON format



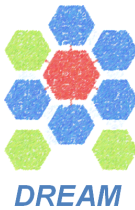
The screenshot shows a dialog box titled "Dream.Source" with a close button. It contains several input fields and a dropdown menu. The fields are: ID (S1), Name (Parts), Interarrival Time (0.5), Distribution Type (Fixed), Mean (0.5), Standard Deviation, Minimum Value, Maximum Value, and Entity Class (Dream.Part). The dialog is overlaid on a background showing a node S1 and an edge E1.

Field	Value
ID	S1
Name	Parts
Interarrival Time	0.5
Distribution Type	Fixed
Mean	0.5
Standard Deviation	
Minimum Value	
Maximum Value	
Entity Class	Dream.Part

Properties of nodes

```
{
  "node": {
    "S1": {
      "name": "Parts",
      "_class": "Dream.Source",
      "interarrivalTime": {
        "distributionType": "Fixed",
        "mean": 0.5
      },
      "entity": "Dream.Part"
    },
    "M1": {
      "name": "M1"
    },
    "E1": {
      "name": "E1"
    }
  },
  "edge": {
    "edge1": {
      "source" : "S1",
      "destination" : "M1",
    },
    "edge2": {
      "source" : "M1",
      "destination" : "E1",
    }
  }
}
```


Introduction to JSON format



- What are the editable properties ?
- Which value type (text, number) ?
- What possible values ?
- What default value ?

Dream.So... ✕

ID

Name

Interarrival Time
Distribution Type
Fixed ▼

Mean

Standard Deviation

Minimum Value


Maximum Value

Entity Class



Introduction to JSON format


Properties are defined as JSON Schema

Dream.So... 

ID

Name

Interarrival Time

Distribution Type
Fixed 

Mean

Standard Deviation

Minimum Value

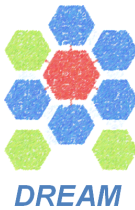
Maximum Value

Entity Class

```
"class_definition": {
  "Dream.Source" : {
    "properties" : {
      "id" : {
        "type" : "string",
        "default" : "S",
        "required" : true,
        "name": "ID",
        "description": "The ID of this source"
      },
      "name" : {
        "type" : "string",
        "default" : "Source"
      },
      "interarrivalTime" : {
        "type": "object",
        "properties": {
          "distributionType": {
            "type": "string",
            "enum": [ "Fixed",
              "Normal",
              "Exponential"],
            "default": "Fixed"
          },
          "mean": {
            "type": "number",
            "default": 0.5
          }
        }
      }
    }
  }
}
```



Introduction to JSON format



Dream.So... ✕

ID

Name

Interarrival Time
Distribution Type
Fixed ▼

Mean

Standard Deviation

Minimum Value

Maximum Value

Entity Class

```
"class_definition": {
  "Dream.Source" : {
    "properties" : {
      "id" : {
        "type" : "string",
        "default" : "S",
        "required" : true,
        "name": "ID",
        "description": "The ID of this source"
      },
      "name" : {
        "type" : "string",
        "default" : "Source"
      },
      "interarrivalTime" : {
        "type": "object",
        "properties": {
          "distributionType": {
            "type": "string",
            "enum": [ "Fixed",
                      "Normal",
                      "Exponential"],
            "default": "Fixed"
          },
          "mean": {
            "type": "number",
            "default": 0.5
          }
        }
      }
    }
  }
}
```



Introduction to JSON format



Dream.So... ✕

ID

Name

Interarrival Time
Distribution Type
Fixed ▼

Mean

Standard Deviation

Minimum Value

Maximum Value

Entity Class

```
"class_definition": {
  "Dream.Source" : {
    "properties" : {
      "id" : {
        "type" : "string",
        "default" : "S",
        "required" : true,
        "name": "ID",
        "description": "The ID of this source"
      },
      "name" : {
        "type" : "string",
        "default" : "Source"
      },
      "interarrivalTime" : {
        "type": "object",
        "properties": {
          "distributionType": {
            "type": "string",
            "enum": [ "Fixed",
                      "Normal",
                      "Exponential"],
            "default": "Fixed"
          },
          "mean": {
            "type": "number",
            "default": 0.5
          }
        }
      }
    }
  }
}
```



Introduction to JSON format



Dream.So... ✕

ID

Name

Interarrival Time

Distribution Type

Fixed ▼

Mean

Standard Deviation

Minimum Value

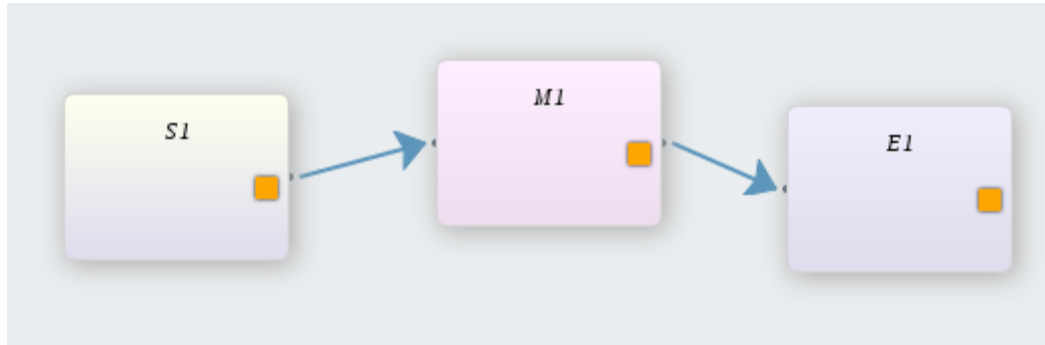
Maximum Value

Entity Class

```
"class_definition": {
  "Dream.Source" : {
    "properties" : {
      "id" : {
        "type" : "string",
        "default" : "S",
        "required" : true,
        "name": "ID",
        "description": "The ID of this source"
      },
      "name" : {
        "type" : "string",
        "default" : "Source"
      },
      "interarrivalTime" : {
        "type": "object",
        "properties": {
          "distributionType": {
            "type": "string",
            "enum": [ "Fixed",
                      "Normal",
                      "Exponential"],
            "default": "Fixed"
          },
          "mean": {
            "type": "number",
            "default": 0.5
          }
        }
      }
    }
  }
}
```



Introduction to JSON format

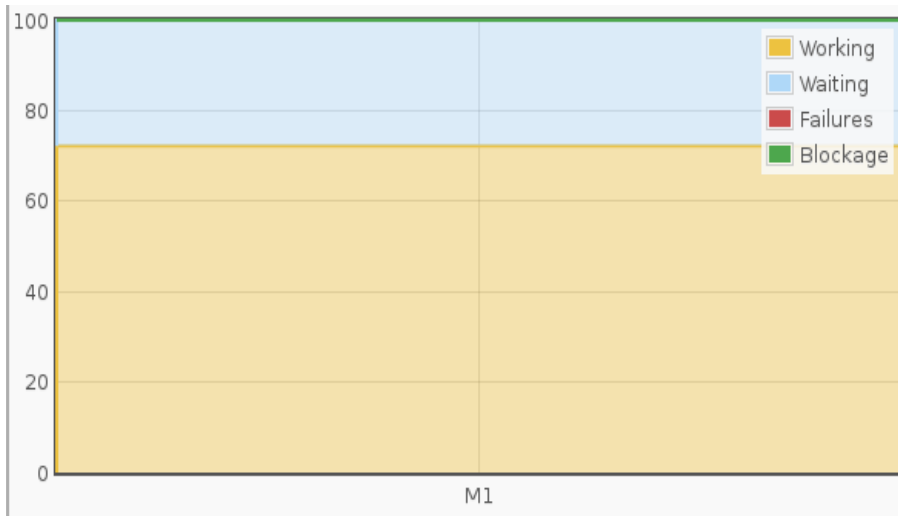
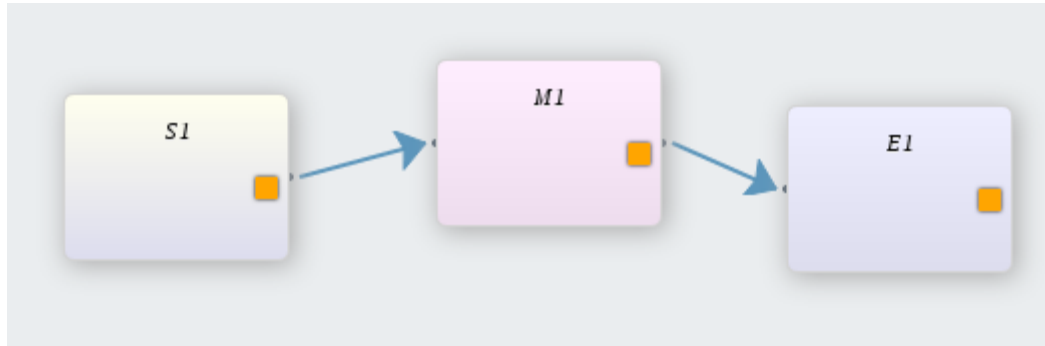


```
{
  "graph": {
    "node": { ... },
    "edge": { ... }
  },
  "class_definition": { ... },
  "input": [
    "WipSpreadsheet" : {
      ["Order 1", "2014/03/15",
"1", "PM1", "Design" ... ],
      ...
    ]
  },
}
```

Order ID	Due Date	Priority	Project Manager	Part	Part Type	Sequence		
Order 1	2014/03/15	1	PM1	Design	Design	CAD		
				Part1	Basic	CAM-MILL-EDM-MILL-MASS		
				Part2	Basic	CAM-MILL-EDM-MILL-EDM-MASS		
				Assemble	Mould	MASS-IM	2-12	
Order 2	2014/03/14	1	PM1	Design	Design	CAD	6	
				Part1	Basic	CAM-MILL-EDM-MILL-MASS	8-4-2-8-0	
				Part2	Basic	CAM-MILL-EDM-MASS	20-15-8-8-0	
				Assemble	Mould	MASS-IM	1-12	
Order 3	2014/03/15	1	PM1	Design	Design	CAD	6	
				Part1	Basic	CAM-MILL-EDM-MASS	8-4-2-0	
				Part2	Basic	CAM-MILL-EDM-MASS	7-15-8-8-0	
				Assemble	Mould	MASS-IM	1-3	

General purpose tabular inputs in the same JSON

Introduction to JSON format



```
{
  "graph": {
    "node": { ... },
    "edge": { ... }
  },
  "class_definition": { ... },
  "input": { ... },
  "output": {
    "M1": {
      "working_ratio": 0.7,
      "waiting_ratio": 0.3,
      "failure_ratio": 0,
      "bloacage_ratio": 0
    }
  }
}
```

Outputs - results of simulation - are also in the same JSON

Introducing Edit Configuration tab



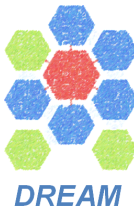
Live edition of JSON

Edit Configuration	
Machine Shifts Spreadsheet	
Repairman Shifts Spreadsheet	
Run Simulation	

Save ↻

```
6      "default": 0.95,
7      "description": "Confidence level for statistical analysis of stochastic experiments",
8      "title": "Confidence level",
9      "type": "number"
10    },
11    "currentDate": {
12      "default": "2014/10/01 09:00:00",
13      "description": "The day the experiment starts, in YYYY/MM/DD HH:MM:SS format",
14      "title": "SimulationStartTime",
15      "type": "string"
16    },
17    "ke_url": {
18      "default": "http://git.erp5.org/gitweb/dream.git/blob_plain/HEAD:/dream/KnowledgeExtraction/Mockup",
19      "description": "The URL for knowledge extraction to access its data for example http://git.erp5.org",
20      "title": "URL for Knowledge Extraction Spreadsheet",
21      "type": "string"
22    },
23    "maxSimTime": {
24      "default": 100,
25      "description": "Length of the simulation run",
26      "title": "Length of Experiment",
27      "type": "number"
28    },
29  }
```


Introducing Edit Configuration tab



Live edition of JSON

Save ↻

```
1 {
2   "application_configuration": {
3     "general": {
4       "properties": {
5         ...},
6         -----^
7         Expecting 'STRING', got 'undefined'
8       }
9     }
10  }
11  }
12  }
13  }
14  }
15  }
16  "currentDate": {
17    "default": "2014/10/01 09:00:00",
18    "description": "The day the experiment st
19    "title": "SimulationStartTime".
```

Parse error on line 10:
...}, ERROR
-----^
Expecting 'STRING', got 'undefined'

for stat

ERROR

Features non obtrusive assistance v

Graph editor: adding a property on node



On Edit configuration tab:

Locate the json schema definition of Dream.Queue in *class_definition*

```
1 ▼ {
2 ►   "application_configuration": {↔},
254 ▼ "class_definition": {
255 ►   "Dream.Edge": {↔},
264 ►   "Dream.EventGenerator": {↔},
329 ►   "Dream.Exit": {↔},
358 ►   "Dream.Machine": {↔},
433 ▼   "Dream.Queue": {
434     "_class": "node",
435     "allOf": [
436 ▼       {
437         "$ref": "#/node"
438       },
439 ▼     ],
440 ▼     "properties": {
441 ▼       "capacity": {
442         "$ref": "#/definitions/_capacity",
443         "required": true
444       },
445 ▼       "id": {
446         "default": "Q",
447         "type": "string"
448       },
449     },
450   },
451 }
```


Graph editor: adding a property on node



On Edit configuration tab:

Add a new property, in this example it is called “Sample property”

```
264 ▶ "Dream.EventGenerator": {↔},
329 ▶ "Dream.Exit": {↔},
358 ▶ "Dream.Machine": {↔},
433 ▼ "Dream.Queue": {
434   "_class": "node",
435   "allof": [
436     {
437       "$ref": "#/node"
438     },
439   ],
440   "properties": {
441     "sample_property": {
442       "name": "Sample Property",
443       "description": "The tooltip to display.",
444       "type": "string"
445     },
446     "capacity": {
447       "$ref": "#/definitions/_capacity",
448       "required": true
449     },
450     "id": {
451       "default": "Q",
452       "type": "string"
453     }
454   }
455 }
```


Graph editor: adding a property on node

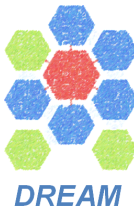


On “Production Line” tab, we have a new property when editing a Dream.Queue.

Set value “user value” and save

A screenshot of a software interface showing a dialog box titled "Dream.Qu...". The dialog box contains several input fields and a dropdown menu. The "id" field is labeled "id" and contains the text "Dream.Queue1". The "name" field is labeled "name" and contains the text "Queue". The "Sample Property" field is labeled "Sample Property" and is empty. The "Capacity" field is labeled "Capacity" and contains the number "1". The "Scheduling Rule" field is labeled "Scheduling Rule" and has a dropdown menu showing "FIFO". At the bottom of the dialog box are two buttons: "Delete" and "Validate".

Graph editor: adding a property on node

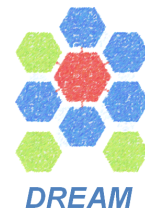


Go back to “Edit configuration” tab, we can see that the sample_property is saved as a property on the node.

Search: `sample_property` (Use /re/ syntax for regexp search)

```
254 ▶ application_configuration : {↔},
916 "class_definition": {↔},
917 ▶ "constraints": {},
928 ▼ "general": {↔},
929 ▶ "graph": {
961 ▼   "edge": {↔},
962 ▶   "node": {
978 ▼     "Dream.Source1": {↔},
979 ▼     "Dream.Queue1": {
980       "coordinate": {
981         "left": 0.26706827309236947,
982         "top": 0.3948110549926837
983       },
984       "_class": "Dream.Queue",
985       "name": "Queue",
986       "id": "Dream.Queue1",
987       "sample_property": "user value",
988       "capacity": 1,
989       "schedulingRule": "FIFO"
990 ▼     },
991 ▼     "Dream.Machine1": {
      "coordinate": {
```


Application configuration: adding a tab



On Edit configuration tab:

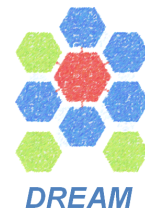
Locate the input action definition in *application_definition*

input is for input action

output is for results visualisation

```
1▼ {
2▼   "application_configuration": {
3▶     "general": {↔},
73▼     "input": {
74▼       "debug": {
75         "gadget": "Input_viewDebugJson",
76         "title": "Edit Configuration",
77         "type": "object_view"
78       },
79▼       "view": {
80         "gadget": "Input_viewProductionLine",
81         "title": "Production Line",
82         "type": "object_view"
83       },
84▼       "view_machine_shift_spreadsheet": {
85▼         "configuration": {
86▼           "columns": [
87▼             {
88               "format": "date-time",
89               "name": "Date",
90               "type": "string"
91             },
92▼           ]
93▼         }
94▼       }
95▼     }
96▼   }
97▼ }
```


Application configuration: adding a tab



Add an action:

gadget: The gadget to use

title: The name to display on the tab

type: Always object_view for now

configuration: Gadget configuration, passed to gadget constructor.

columns: columns definition

input_id: key for this input

```
70 }
71 }
72 }
73 ▼
74 ▼
75     "input": {
76         "sample_spreadsheet": {
77             "gadget": "Input_viewSpreadsheet",
78             "title": "Sample Spreadsheet",
79             "type": "object_view",
80             "configuration": {
81                 "input_id": "sample_spreadsheet_input",
82                 "columns": [
83                     {
84                         "name": "Sample Column",
85                         "type": "string"
86                     }
87                 ]
88             }
89         }
90     },
91     "debug": {
92         "gadget": "Print_and_Debug"
93     }
94 }
```


Application configuration: adding a tab

The new tab appears. Set some values

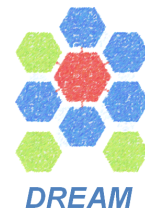
Sample Spreadsheet
Production Line
Manage document
Results
WIP Spreadsheet
Sample Column
value 1
value 2
value 3

Application configuration: adding a tab

The values are saved in the JSON, as input / (the input_id configured)

```
267 ▶ "class_definition": {↔},
929 "constraints": {},
930 ▶ "general": {↔},
941 ▶ "graph": {↔},
1079 ▼ "input": {
1080 ▶   "repairman_shift_spreadsheet": [↔],
1094 ▶   "wip_spreadsheet": [↔],
1106 ▶   "machine_shift_spreadsheet": [↔],
1130 ▼   "sample_spreadsheet_input": [
1131 ▼     {
1132       "name": "Sample Column"
1133     },
1134 ▼     {
1135       "name": "value 1"
1136     },
1137 ▼     {
1138       "name": "value 2"
1139     },
1140 ▼     {
1141       "name": "value 3"
1142     },
1143 ▼     {
1144       "name": null
```


Application configuration: adding a tab



To add a completely new gadget, locate the gadget folder in dream-repository.
git/dream/platform/src/dream

A screenshot of a code editor interface. On the left, a file explorer shows the directory structure: dream-repository.git > dream > platform > src > dream. The file 'Input_viewAttachDocu' is selected. The main editor area shows the code for this file. The code starts with a function 'initGadgetMixin' that uses 'use strict' and initializes a 'gadget_klass' with 'rJS(window)'. It then declares two methods: 'aq_putAttachment' and 'aq_getAttachment'. The 'startService' method is declared, which creates a button and sets up event listeners for 'attach_document' and 'ui-btn'. The 'attach_document' listener calls 'push' with a function that returns a 'promiseEventListener' for the 'attach_document' event. The 'ui-btn' listener sets 'button.disabled' to 'true' when clicked.

```
4 initGadgetMixin() {
5   "use strict";
6
7   var gadget_klass = rJS(window);
8   initGadgetMixin(gadget_klass);
9   gadget_klass
10
11   // Acquired methods
12   // Acquired methods
13   .declareAcquiredMethod("aq_putAttachment", "jio_putAttachment")
14   .declareAcquiredMethod("aq_getAttachment", "jio_getAttachment")
15
16   // declared methods
17   // declared methods
18   // declared methods
19   .declareMethod("startService", function () {
20     var gadget = this,
21       encoded_data,
22       button;
23     return new RSVP.Queue()
24       .push(function () {
25         return promiseEventListener(
26           gadget.props.element.getElementsByClassName("attach_document")[0],
27           'submit',
28           false
29         );
30       })
31       .push(function (evt) {
32         button = evt.target.getElementsByClassName("ui-btn")[0];
33         button.disabled = true;
34       });
35   });
36 }
```


Application configuration: adding a tab

- Clone existing gadget:
 - html file, usually Input_viewXXXGadget.html
 - javascript file, usually Input_viewXXXGadget.js
- Make sure html file reference the javascript properly
- Debugging tips:
 - code you edit is in dream/platform/src/
 - grunt compiles code in dream/platform/static
 - if syntax error, files in static are not updated !
 - static files have different lines numbers
- use console.log (better with firebug)
- <http://learn.renderjs.org/docs/index.html>



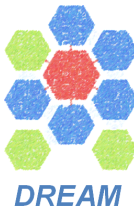
Pre/Post Processing plugins

JSON data goes through :

- pre processing plugins
- simulation engine
- post processing plugins



Pre/Post Processing plugins



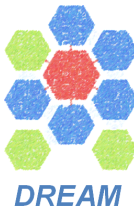
Plugins configured in JSON, under *application_configuration*

```
1▼ {
2▼   "application_configuration": {
3▶     "general": {↔},
73▶     "input": {↔},
186▶     "output": {↔},
231▼     "post_processing": {
232       "description": "",
233       "plugin_list": [
234         {
235           "_class": "dream.plugins.DefaultTabularExit.DefaultTabularExit"
236         },
237         {
238           "_class": "dream.plugins.PostProcessStationUtilization.PostProcessStationUtilization",
239           "family": "Server",
240           "output_id": "station_utilization"
241         },
242         {
243           "_class": "dream.plugins.PostProcessQueueStatistics.PostProcessQueueStatistics",
244           "output_id": "queue_statistics"
245         }
246       ]
247     },
248     "pre_processing": {
249       "description": ""
250     }
251   }
252 }
```

Python class for plugin

Parameters for the plugin

Pre/Post Processing plugins

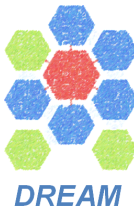


Plugins are python class, interface is `postprocess(self, data)` or `preprocess(self, data)`

A screenshot of a code editor. On the left, a file explorer shows a list of Python files: JSStationUtilization.py, MergeSteps.py, OldStylePartJobShopWIP.py, ParseTraceFile.py, plugin.py, PostProcessDemandPlanning.py, PostProcessOrderLateness.py, PostProcessQueueStatistics.py, PostProcessStationUtilization.py (highlighted), PrepareExampleGantt.py, ReadEntryData.py, ReadJSCompleted.py, ReadJSOrders.py, ReadJSShifts.py, ReadJSSkills.py, ReadJSSkillsToStations.py, and ReadJSWIP.py. On the right, the code for PostProcessStationUtilization.py is displayed. The code defines a class PostProcessStationUtilization that inherits from plugin.OutputPreparationPlugin. It has a docstring and a postprocess method. The postprocess method takes self and data as arguments, extracts data from the 'result' list, and sets up a plot with ticks, working_data, waiting_data, failure_data, blockage_data, and setup_data. The plot options are defined in the options dictionary, including xaxis, yaxis, series, and bars. The code is as follows:

```
1 from dream.plugins import plugin
2
3 class PostProcessStationUtilization(plugin.OutputPreparationPlugin):
4     """ Output the station utilization metrics in a format compatible with
5     """
6
7     def postprocess(self, data):
8         result = data['result']['result_list'][-1]
9
10        ticks = []
11        working_data = []
12        waiting_data = []
13        failure_data = []
14        blockage_data = []
15        setup_data = []
16
17        options = {
18            "xaxis": {
19                "minTickSize": 1,
20                "ticks": ticks
21            },
22            "yaxis": {
23                "max": 100
24            },
25            "series": {
26                "bars": {
27                    "show": True,
28                    "barWidth": 0.8,
29                    "align": "center"
```


Pre/Post Processing plugins



Plugin configuration defined in json is in `self.configuration_dict`

```
49         "label": "Setup",
50         "data": setup_data
51     });
52
53     out = result[self.configuration_dict['output_id']] = {
54         "series": series,
55         "options": options
56     }
57
58     i = 0
59     for obj in result['elementList']:
60         if obj.get('family') == self.configuration_dict.get('family'):
61             if obj['results']['working_ratio']:
62                 working_data.append((i, obj['results']['working_ratio'][0]))
63             if obj['results']['waiting_ratio']:
64                 waiting_data.append((i, obj['results']['waiting_ratio'][0]))
65             if obj['results']['failure_ratio']:
66                 failure_data.append((i, obj['results']['failure_ratio'][0]))
67             if obj['results']['blockage_ratio']:
68                 blockage_data.append((i, obj['results']['blockage_ratio'][0]))
69             if obj['results']['setup_ratio']:
70                 setup_data.append((i, obj['results']['setup_ratio'][0]))
71
72             ticks.append((i, obj.get('name', obj['id'])))
73             i += 1
74
75     return data
```


Pre/Post Processing plugins

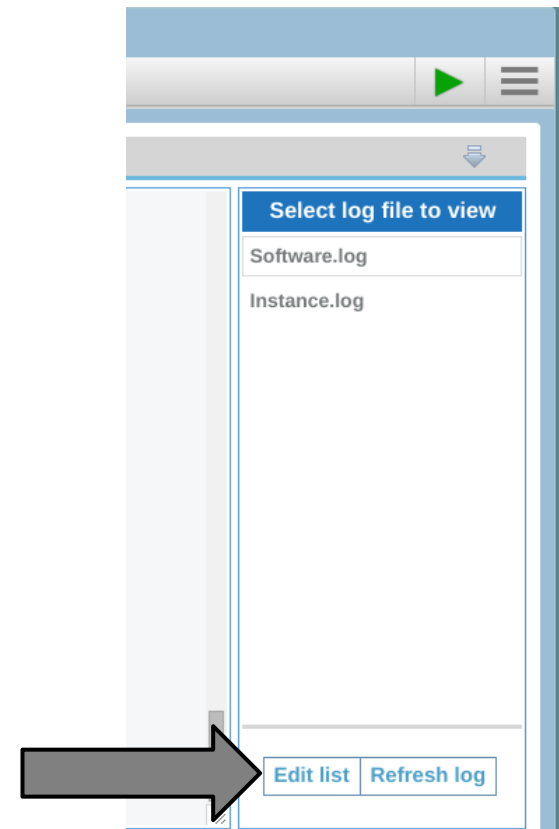
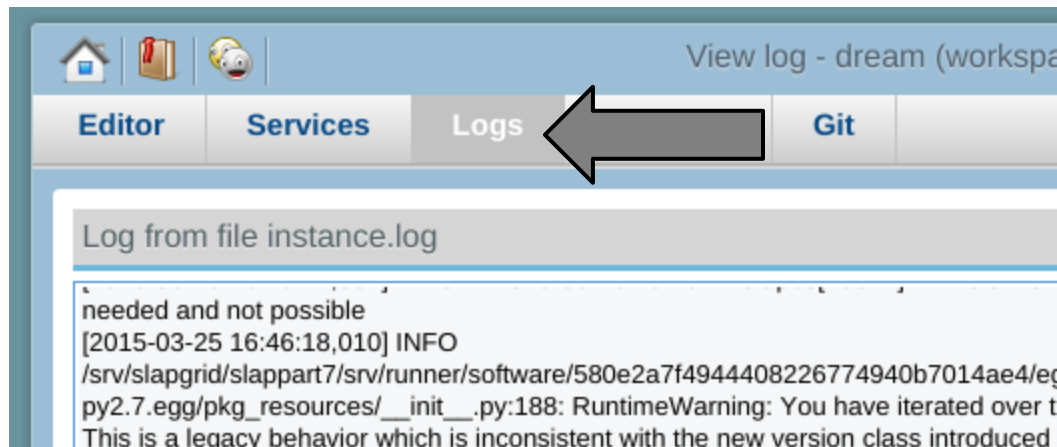


Plugins debugging tips:

- Use existing Knowledge Extraction Tool objects (from dream/KnowledgeExtraction/ folder)
- Use dream.plugin.TimeSupport to convert from and to simulation clock time
- Log using self.logger.info (standard python logger <https://docs.python.org/2/library/logging.html>)
- Use dream.plugin.Debug in the chain to dump the json
- Write unit tests (examples in dream/tests/testGUIModels.py)

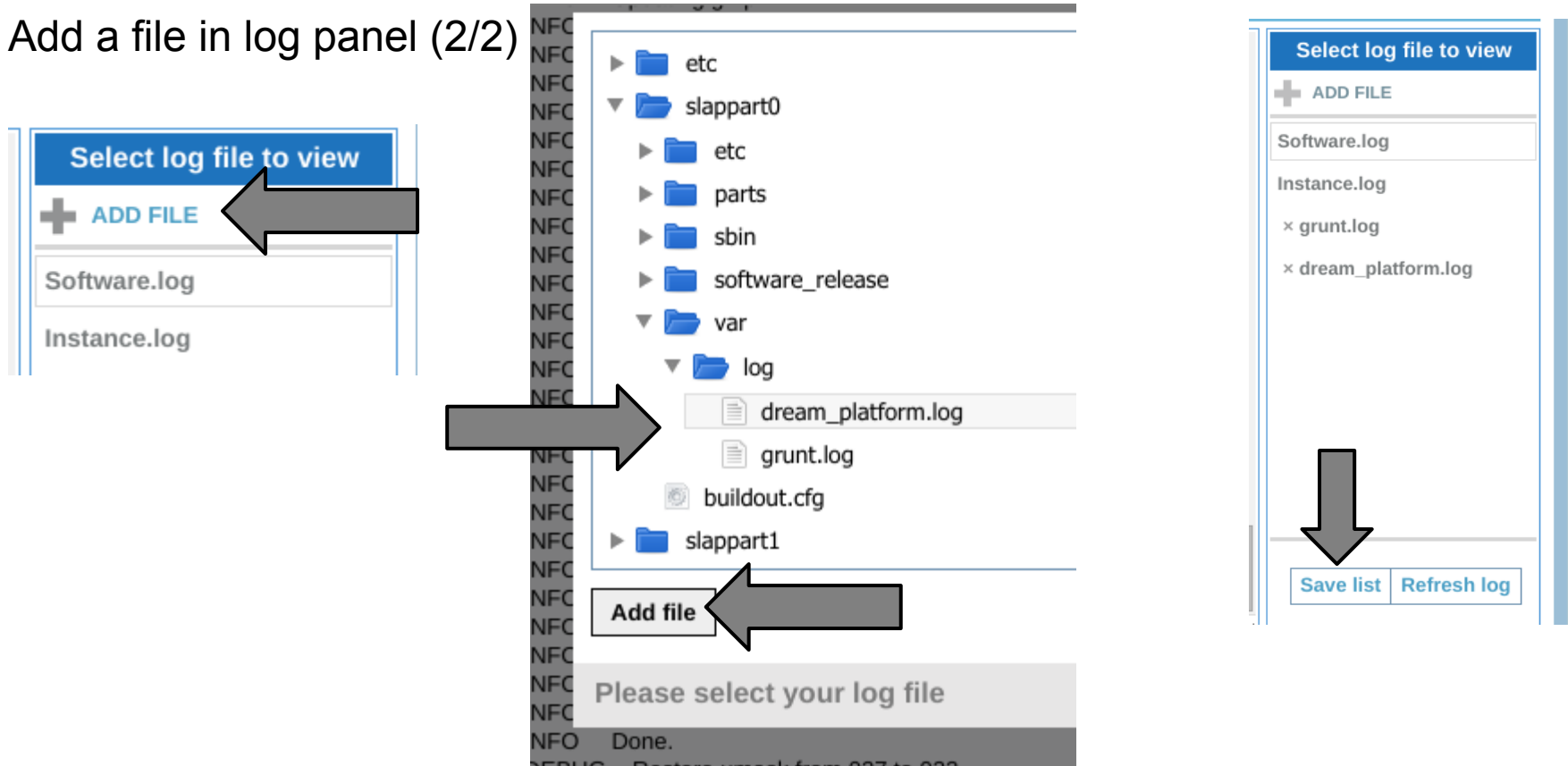
Accessing the application logs

Add a file in log panel (1/2)



Accessing the application logs

Add a file in log panel (2/2)



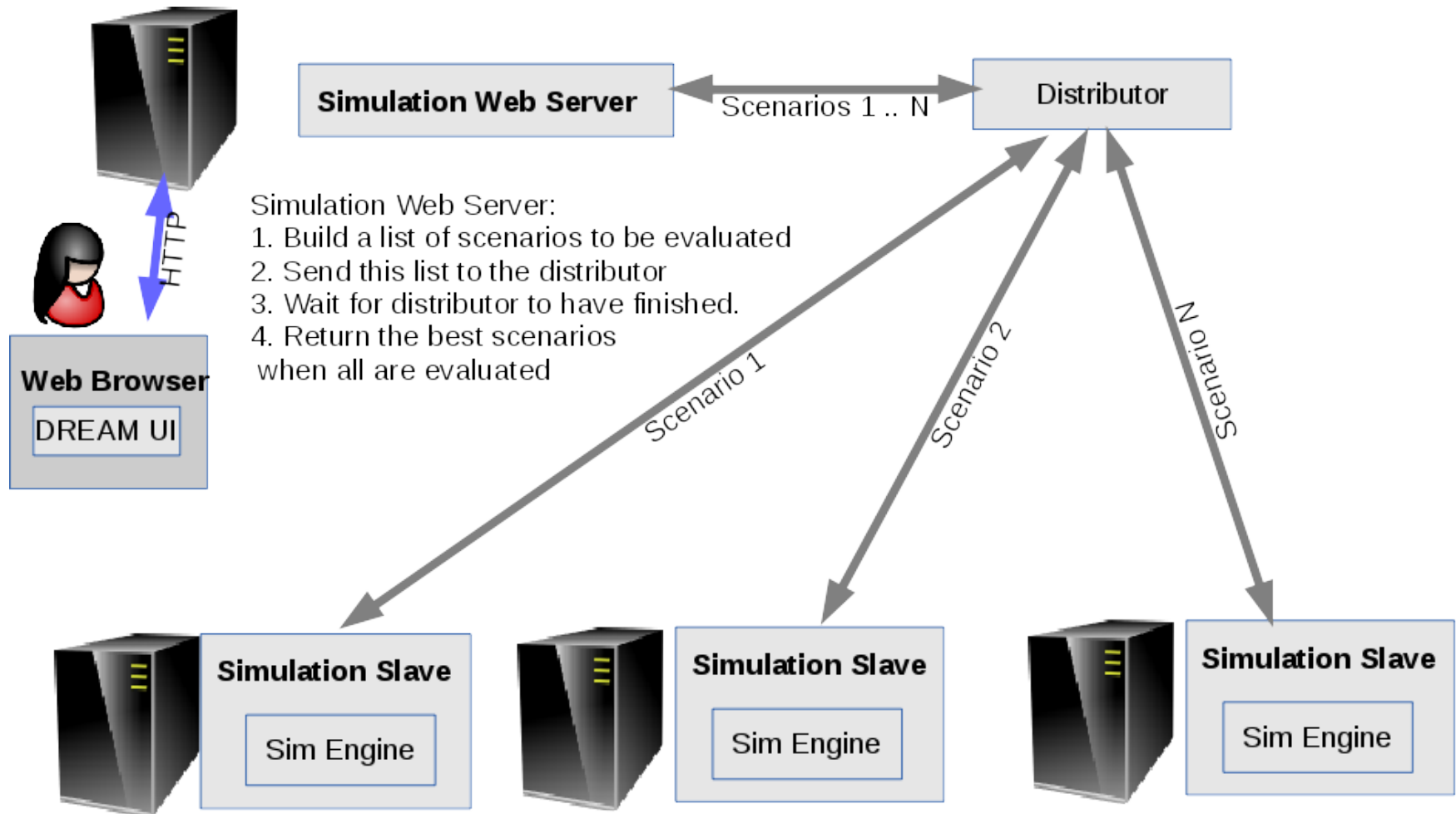
The image shows a sequence of three screenshots illustrating the process of adding a log file to the application log panel.

Left Screenshot: A panel titled "Select log file to view" with an "ADD FILE" button. Below the button, a list of log files is shown: "Software.log" and "Instance.log". A large grey arrow points from the "ADD FILE" button to the central file explorer.

Center Screenshot: A file explorer window showing a directory structure. The "log" folder is expanded, showing files including "dream_platform.log", "grunt.log", and "buildout.cfg". A large grey arrow points from the "dream_platform.log" file to the "Add file" button at the bottom.

Right Screenshot: The "Select log file to view" panel again, but now with a list of log files including "Software.log", "Instance.log", "× grunt.log", and "× dream_platform.log". A large grey arrow points from the "Save list" button at the bottom to the final state of the panel.

Cloud Execution



- Install DREAM distributor http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/heads/dream:/software/erp5/software.cfg
- Note the custom frontend URL for later reference
- Install some DREAM simulation nodes http://git.erp5.org/gitweb/slapos.git/blob_plain/refs/heads/dream:/software/testnode/software.cfg set the distributor URL as “test-suite-master-url” parameter. Or using the following XML Parameter for web runner:

```
<instance>
  <parameter id="parameter-test-node-title">Simulation Node 1</parameter>
  <parameter id="parameter-test-suite-master-url">https://login:
password@softinstXXX.host.vifib.
net/erp5/portal_task_distribution/dream_distributor</parameter>
  <parameter id="slapos-software">software/erp5testnode</parameter>
  <parameter id="slapos-reference">dream</parameter>
</instance>
```


Be sure to use dream.plugin.ACO.ACO as processing plugin class in your model

```
763 ▶ "output": {↔},
839 ▼ "pre_processing": {
840 ▼   "plugin_list": [
841 ▼     {
842       "_class": "dream.plugins.Debug.Debug",
843       "argument": "Argument Value"
844     },
845 ▼     {
846       "_class": "dream.plugins.OldStylePartJobShopWIP.OldStylePartJobShopWIP",
847       "input_id": "old_style_part_jobshop_spreadsheet"
848     }
849   ]
850 },
851 ▼ "processing_plugin": {
852   "_class": "dream.plugin.ACO.ACO"
853 },
854 ▼ "post_processing": {
855   "plugin_list": []
856 },
857 ▼ "general": {
858 ▼   "properties": {
859 ▼     "numberOfReplications": {
860       "name": "Number of replications"
```



Be sure to specify the distributor URL in general properties

Seed for random number generator

Number of generations

Number of ants per generation

Number of solutions

Distributor URL



Number of simulation nodes

View progress in “Test Result Module” of the distributor

ERP5 / Test Results / DREAM Simulation Run / Logged in as : dream

View Nodes Graph Consistency History Metadata

Title	DREAM Simulation Run	Project	<input type="text"/>
Reference	SIMPAT_20_orders_20_ants_Deterministic	All Tests	0
Test Report ID	928	Failures	0
Launch Date	2015/01/25 22:18	Errors	0
Completion Date	2015/01/25 22:20	Skips	0
		Test Result	PASS
		Test Status	Completed

Comment

Total processing time: 138.312806
 Total execution time (including protocol): 312.004467
 Total execution time (only manpy): 9.2277841568

Test Results : 1 - 0 of 20 records

Test Case	Start Date	Duration	Manpy Execution Time	Status
{u'QIM': 'EDD', u'QStart': 'RPC', u'QMASS': 'Priority', u'QMILL': 'SPT', u'QCAM': 'FIFO', u'QEDM': 'Priority', u'QTURN': 'SPT'}	2015/01/25 22:19:42.885798 UTC	15.208942	0.489723920822	Completed
{u'QIM': 'EDD', u'QStart': 'SPT', u'QMASS': 'WINQ', u'QMILL': 'SPT', u'QCAM': 'EOD', u'QEDM': 'NumStages', u'QTURN': 'WINQ'}	2015/01/25 22:19:45.376398 UTC	14.172767	0.328832149506	Completed
{u'QIM': 'EDD', u'QStart': 'WINQ', u'QMASS': 'LPT', u'QMILL': 'EOD', u'QCAM': 'RPC', u'QEDM': 'WINQ', u'QTURN': 'EOD'}	2015/01/25 22:19:25.806784 UTC	15.131353	0.496667146683	Completed
{u'QIM': 'EOD', u'QStart': 'Priority', u'QMASS': 'EDD', u'QMILL': 'EDD', u'QCAM': 'EOD', u'QEDM': 'NumStages', u'QTURN': 'RPC'}	2015/01/25 22:20:1.496204 UTC	14.408654	0.369523048401	Completed
{u'QIM': 'FIFO', u'QStart': 'MS', u'QMASS': 'NumStages', u'QMILL': 'Priority', u'QCAM': 'EOD', u'QEDM': 'EOD', u'QTURN': 'RPC'}	2015/01/25 22:18:51.861788 UTC	15.180787	0.495496034622	Completed
{u'QIM': 'LPT', u'QStart': 'Priority', u'QMASS': 'Priority', u'QMILL': 'FIFO', u'QCAM': 'EOD', u'QEDM': 'LPT', u'QTURN': 'EDD'}	2015/01/25 22:19:26.947794 UTC	16.429031	0.550364017487	Completed
{u'QIM': 'LPT', u'QStart': 'SPT', u'QMASS': 'RPC', u'QMILL': 'MS', u'QCAM': 'Priority', u'QEDM': 'FIFO', u'QTURN': 'Priority'}	2015/01/25 22:18:50.794476 UTC	16.041735	0.555281162262	Completed
{u'QIM': 'MS', u'QStart': 'MS', u'QMASS': 'EOD', u'QMILL': 'RPC', u'QCAM': 'MS', u'QEDM': 'Priority', u'QTURN': 'WINQ'}	2015/01/25 22:20:5.955112 UTC	16.498534	0.573115825653	Completed
{u'QIM': 'NumStages', u'QStart': 'NumStages', u'QMASS': 'SPT', u'QMILL': 'NumStages', u'QCAM': 'MS', u'QEDM': 'Priority', u'QTURN': 'RPC'}	2015/01/25 22:20:24.754760 UTC	16.344587	0.552254915237	Completed
{u'QIM': 'Priority', u'QStart': 'MS', u'QMASS': 'LPT', u'QMILL': 'EOD', u'QCAM': 'WINQ', u'QEDM': 'SPT', u'QTURN': 'EDD'}	2015/01/25 22:18:50.853945 UTC	15.010426	0.339298009872	Completed
{u'QIM': 'RPC', u'QStart': 'RPC', u'QMASS': 'WINQ', u'QMILL': 'RPC', u'QCAM': 'MS', u'QEDM': 'EDD', u'QTURN': 'RPC'}	2015/01/25 22:20:18.677495 UTC	14.220671	0.325212001801	Completed
{u'QIM': 'RPC', u'QStart': 'MS', u'QMASS': 'NumStages', u'QMILL': 'MS', u'QCAM': 'FIFO', u'QEDM': 'WINQ', u'QTURN': 'Priority'}	2015/01/25 22:19:8.728046 UTC	16.402627	0.558176994324	Completed
{u'QIM': 'RPC', u'QStart': 'RPC', u'QMASS': 'WINQ', u'QMILL': 'RPC', u'QCAM': 'MS', u'QEDM': 'EDD', u'QTURN': 'FIFO'}	2015/01/25 22:19:45.312931 UTC	18.448665	0.552452087402	Completed
{u'QIM': 'RPC', u'QStart': 'WINQ', u'QMASS': 'RPC', u'QMILL': 'MS', u'QCAM': 'SPT', u'QEDM': 'LPT', u'QTURN': 'EDD'}	2015/01/25 22:19:8.876556 UTC	15.107798	0.485086917877	Completed
{u'QIM': 'SPT', u'QStart': 'LPT', u'QMASS': 'MS', u'QMILL': 'WINQ', u'QCAM': 'SPT', u'QEDM': 'RPC', u'QTURN': 'LPT'}	2015/01/25 22:20:34.908766 UTC	14.25004	0.331388950348	Completed
{u'QIM': 'SPT', u'QStart': 'NumStages', u'QMASS': 'LPT', u'QMILL': 'LPT', u'QCAM': 'EOD', u'QEDM': 'MS', u'QTURN': 'EOD'}	2015/01/25 22:20:35.121873 UTC	15.211308	0.46063095665	Completed

Cloud execution only has benefit for long running scenarios

