

SHIWA Repository Admin Manual

This manual documents the SHIWA application repository. Sections 1-2 describe the key entities, actors and use cases, section 3-13 describe how the provided features can be accessed via the GUI.

TABLE OF CONTENTS

1. Introduction	5
Background	5
Project goal.....	5
Key Components	5
Links	5
How to register as a user.....	6
2. Entity, Actor & Use-case Specification	6
2.1 Entity Definitions	6
2.2 Actor Definitions.....	6
2.3 Repository Model.....	7
2.4 Use Cases	7
E-scientist.	7
Workflow Developer.	8
Repository Administrator.....	9
Validator.....	10
3. GUI Structure	11
4. Manage User Profile.....	11
4. Workflow management.....	13
4.1 List and search validated workflows	13
4.1.1 Workflow Browse view	13
4.1.2 Workflow Table view (Figure 10).....	13
4.1.3 Workflow details (Figure 11).....	14

4.1.4 Workflow Attributes (Figure 12).....	15
4.1.5 Workflow files & Download (Figure 13)	16
4.1.6 Workflow Implementations (Figure 14).....	17
4.2 Create Workflows	17
4.3 Modify/Delete workflow	18
4.3.1 Ownership	18
4.3.2 Access Control	19
4.3.3 Attributes	20
4.3.4 Files	20
4.3.5 Implementations	21
5. Implementation management.....	22
5.1 List and search implementations	22
5.1.1 Implementation Browse view.....	22
5.1.2 Implementation Table view.....	22
5.1.3 Implementations details (Figure 25)	23
5.1.4 Implementation Attributes.....	24
5.1.5 Implementation files & Download	25
5.2 Create Implementation	26
5.3 Modify Implementation.....	26
5.3.1 Details	26
5.3.2 Attributes	27
5.3.3 Files	28
5.3.4 Graph images of implementations.....	28
5.3.5 Validation	29
5.3.6 Workflow Execution.....	29
6. Import Workflows from MyExperiment.....	33
6.1 Preface	33
6.2 Importing Workflows	33

6.3 Input Port Configuration	34
7. Workflow Execution extension	33
7.1 Preface	33
7.2 Create execution.....	33
7.3 Configure parameters	34
7.4 Deploy implementation to the GEMLCA Service	37
7.6 Modify execution.....	38
7.7 Execution of prototype workflow from the Appendix of the Usage Policy	38
8. Implementation Life Cycle	39
9. Workflow and Implementation Validation	40
9.1 Owner	40
9.2 Validator	40
10. User management.....	42
10.1 Browse users.....	42
10.2 Create users	42
10.3 Modify users	42
10.4 Delete users.....	43
11. Group management	44
11.1 Browse groups.....	44
11.2 Create groups.....	44
11.3 Modify groups	44
11.4 Delete groups	44
12. Engine management	45
12.1 Browse engines	45
12.2 Create Engines.....	45
12.3 Modify engines.....	45
12.4 Delete engines.....	45

13. Servlet interface	46
13.1 Listing key information about repository items	46
13.2 Signature validation	47
13.3 Download bundle	47
14. Repository metadata graph (Figure 41).....	49
15. Limitations	50

1. INTRODUCTION

BACKGROUND

Researchers of all disciplines, from Life Sciences and Astronomy to Computational Chemistry, create and use ever-increasing amounts of complex data, and rely more and more on compute-intensive modelling, simulation and analysis.

Scientific workflows have become a key paradigm for managing complex tasks and have emerged as a unifying mechanism for handling scientific data. Workflows capture the essence of the scientific process, providing a means to describe it via logical data- or work-flows. Workflows are mapped onto concrete Distributed Computing Infrastructures (DCIs) to perform large-scale experiments.

The learning curve for reusing workflows, however, is still steep because workflows typically have their own user interfaces/APIs, description languages, provenance strategies, and enactment engines, which are not standard and do not interoperable. Workflow integration or reuse therefore is currently impractical, thereby inhibiting the growth in uptake and proliferation of workflows in scientific practice.

PROJECT GOAL

User communities from all around Europe use many kinds of different workflow languages. Communities develop their workflows using one of the workflow engines. Workflow development, testing and validation are time consuming processes and require specific expertise. These limit the number of available workflows, so it is important to reuse them. Workflows developed for one workflow system is normally not compatible with workflows of other workflow systems. In the past if two user communities using different workflow systems wanted to collaborate, they had to create the workflows from scratch to transform them to the desired workflow languages. This situation can be resolved by emerging new workflow interoperability technologies. The goal of SHIWA is to develop such technologies.

According to the new SHIWA technologies publicly available workflows can be used by different research communities working on different workflow systems and are enabled to run on multiple distributed computing infrastructures. As a result workflow communities are not locked anymore in to their selected workflow system and it's supported distributed computing infrastructure.

KEY COMPONENTS

SHIWA develops, deploys and operates the SHIWA Simulation Platform to offer users production-level services supporting workflow interoperability. As part of the SHIWA Simulation Platform the SHIWA Repository facilitates publishing and sharing workflows, and the SHIWA Portal enables their actual enactment and execution in all the DCIs available in Europe. Use cases targeting various scientific domains will serve to drive and evaluate this platform from a user's perspective.

LINKS

SHIWA Workflow Repository – Administrator Manual

The SHIWA homepage is <http://www.shiwa-workflow.eu>

The SHIWA Simulation platform can be found at

<http://shiwa-portal.cpc.wmin.ac.uk/liferay-portal-6.0.5>

The SHIWA Repository can be found at

<http://shiwa-repo.cpc.wmin.ac.uk>

HOW TO REGISTER AS A USER

All the processes described in this user manual can be performed on the Public page by any E-Scientist, without the requirement to login. Should you want to register, to enable workflow development, please send an email to weingan@wmin.ac.uk.

2. ENTITY, ACTOR & USE-CASE SPECIFICATION

2.1 ENTITY DEFINITIONS

Workflow. This entity represents an abstract workflow. It describes the inputs and outputs and explains what the workflow does in an abstract manner, provides example inputs and outputs (configurations), and some further information.

Implementation. This entity represents an implementation of a workflow. It strictly follows the input and output definitions of the workflow and implements the functionality given in the workflow description. It contains or references (via e.g. URLs) all the files and also holds dependencies to run the workflow on its associated workflow engine.

Engine. This entity represents a workflow engine that is able to interpret and execute a given implementation.

User. This entity represents a repository user associated with a user role.

Group. This entity grants read/write/download rights to a particular workflow for a set of users (the members of the group).

Platform. This entity describes in which desktop Grid and/or service Grid environment the implementation can be executed.

Files. This entity contains the files related to workflows and their implementations.

2.2 ACTOR DEFINITIONS

E-scientist. This actor is the consumer of the contents of the repository, i.e. workflow engines and workflows to run their experiments. This actor does not require credentials to login.

Workflow Developer. This actor is the creator and maintainer of the contents of the repository, i.e. workflows and their implementations and configurations.

Repository Administrator. This actor is a system administrator who has the highest role among the actors. His task is to maintain the SHIWA repository and to support all other users.

Validator. This actor is a computer scientist who wants to test workflows and implementations created by workflow developers and give feedback. The actor

should find non-validated workflows submitted by workflow developers, download packages and sample inputs and attempt to run the workflows and implementations. After successful validation he gives a feedback about the workflows and implementations and makes them available for E-scientists by marking them as validated.

2.3 REPOSITORY MODEL

Users represent all actors (e-scientists, workflow developers, validators and repository administrators) see Figure 1. We distinguish among actors based on their roles and the corresponding access rights. Workflow developers may own workflows and their implementations. E-scientists can browse validated workflows and their validated implementations, download and run them on the SHIWA Simulation Platform.

The repository also enables workflow developers and repository administrator to create and manage groups. Groups support controlled access to workflows and their implementations. Workflows have implementations and attributes.

Implementations have attributes and files. They are also associated with workflow engines.

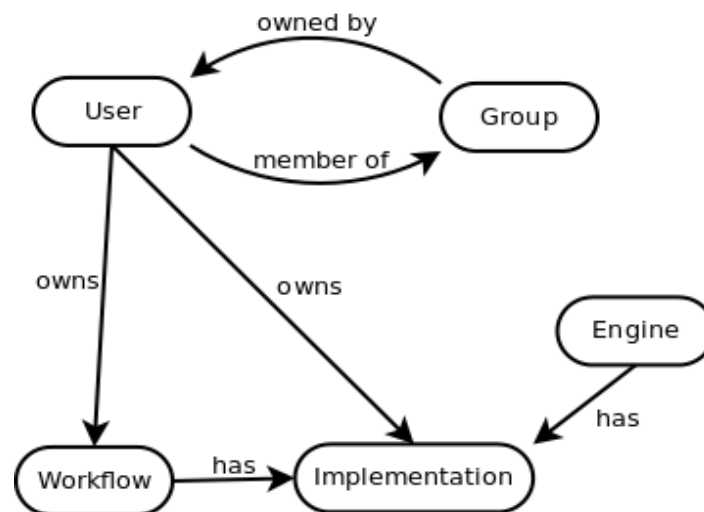


Figure 1: Repository model

2.4 USE CASES

This user manual covers the use cases of all users.

E-SCIENTIST. E-scientists are the consumers of the contents of the repository and can access the following functionality (see Figure 2).

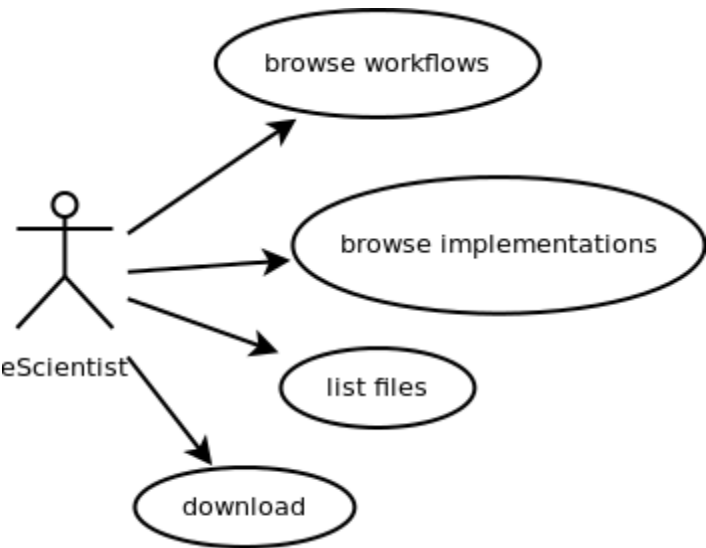


Figure 2: E-Scientist use case

Browse workflows (see section

4. Workflow)

Browsing includes searching and listing workflows based on their metadata.

Browse **implementations** **(see** **section**

5. Implementation management)

E-scientists can browse implementations of the workflows selected by the “Browse workflows” operation.

List files (see section 5.1.5 Implementation files & Download)

E-scientists can list files belonging to workflow implementations selected by the “Browse implementation” operation.

Download (see **sections**

4. Workflow and section

5. Implementation management)

Users can download workflows and their related entities (implementations, configurations and files).

WORKFLOW DEVELOPER. They are the creators and maintainers of the contents of the repository, i.e. workflows and their implementations and configurations.

Manage owned groups

Workflow developers can create user groups. They will own these groups, i.e. they will be the group leaders. They as group leaders can display, modify and delete these groups.

Manage owned workflows

Workflow developers are allowed to upload, modify, delete and download workflows the repository.

Manage owned implementations

Workflow developers are enabled to upload, modify, delete and download workflow implementations.

Mark implementation for validation

Developers are allowed to upload, modify, delete and download files inputs associated to applications

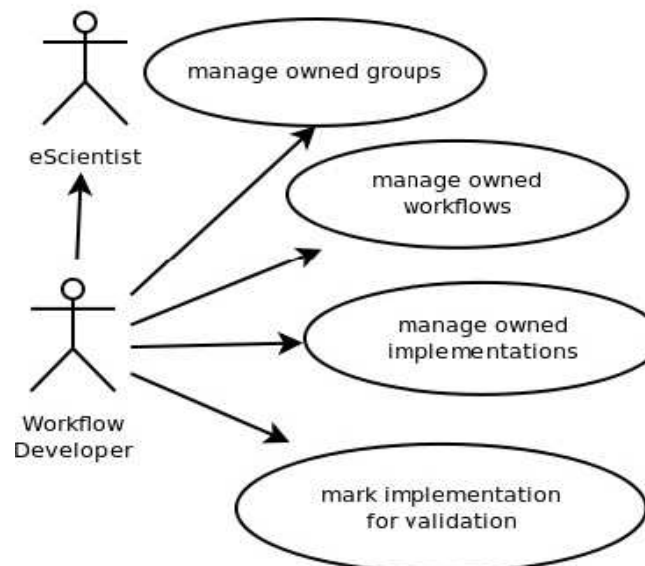


Figure 3: Workflow Developer use case

REPOSITORY ADMINISTRATOR. This actor is a system administrator has the highest role among the actors. His task is to maintain the SHIWA repository and to support all other users.

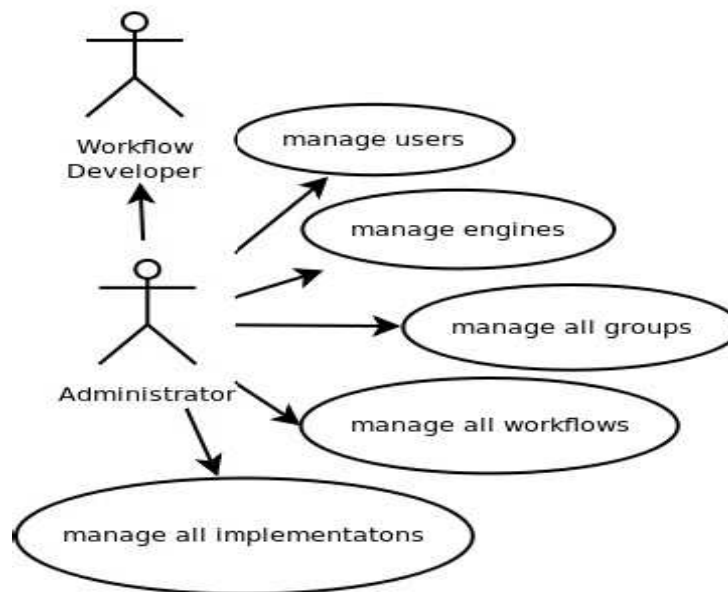


Figure 4: Repository administrator use case

Manage users

Workflow developers can create groups of users. They will own these groups, i.e. they will be the group leaders. They as group leaders can display, modify and delete these groups.

Manage engines

Developers are allowed to upload, modify, delete and download files sample inputs associated to applications.

Manage all groups

Similarly to workflow developers administrators can create user groups. They can display, modify and delete these groups.

Manage workflows

Administrators are allowed to upload workflows into the repository and modify, delete and download all workflow in the repository.

Manage implementations

Administrators are allowed to upload implementations into the repository, modify, delete and download all implementations in the repository.

VALIDATOR. Validators download workflow and implementation packages from the repository marked for validation, do the validation, change the status of the workflow or implementation and give feedback. However they do not provide any other services, their job is only to test/validate workflows and implementations.

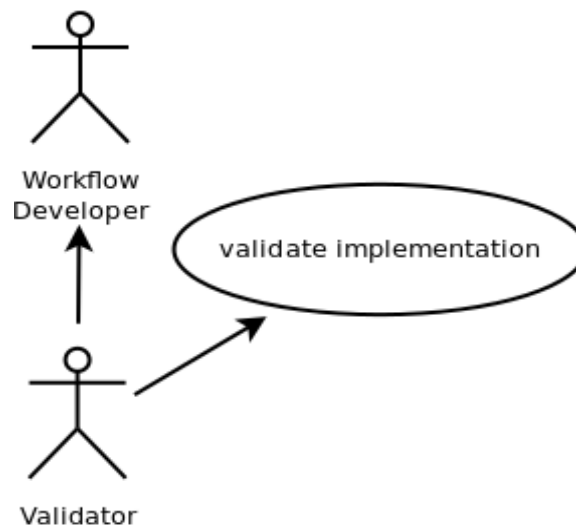


Figure 5: Implementation validator use case

Validate workflows

Validators are allowed to validate workflows, making them visible by E-Scientists. The validation process involves verifying that the abstract workflow definition is complete.

Validate implementations

Validators are allowed to validate implementations marked for validation by the workflow developer. The validation process includes the following operations: download implementations and files needed to run them, and change the status of the implementation from non-validated to validated after a successful validation.

3. GUI STRUCTURE

Repository features can be accessed using the main menu on the top. The following chapters go through the Workflows and Implementations tabs and describe the provided functionality. Information related to the selected tab is displayed in a table. Rows of a table can be filtered by entering text into the text field below any column title. Actions can be initiated using an “Actions” tab on the right. See illustration in Figure 6.

This is the GUI structure of the public view, which can be viewed without any login credentials.

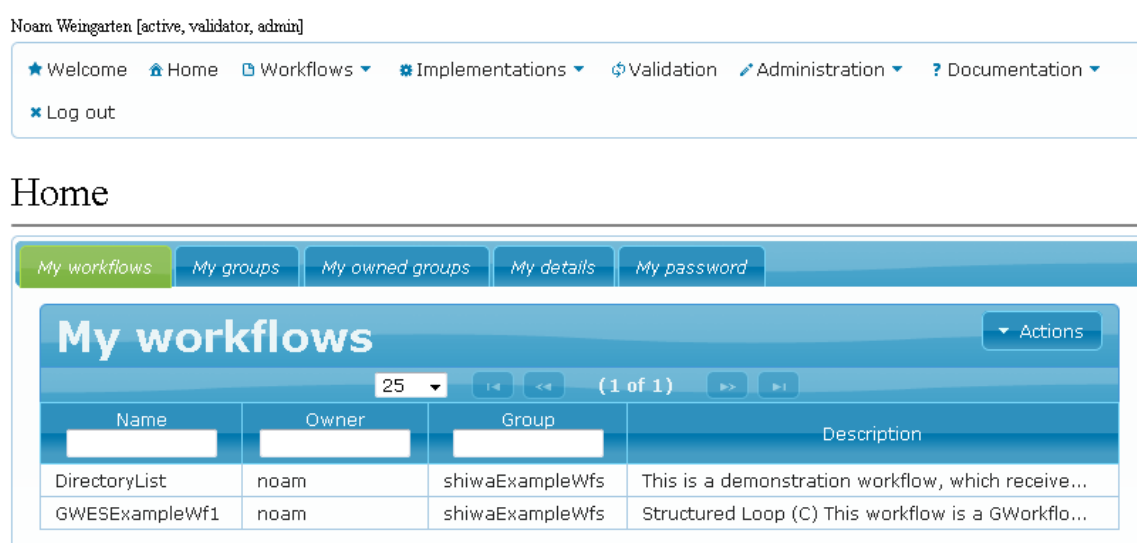


Figure 6 : GUI Structure

This Home view interface can be used to list the user's owned workflows, groups of which the member is a member of and groups the user owns. See section # for workflow management, and section # for Group management.

4. MANAGE USER PROFILE

The *My details* and *My password* tabs from Figure 7 can be used to manage the users details and to change the users password (see Figure 8 and Figure 9)

Home

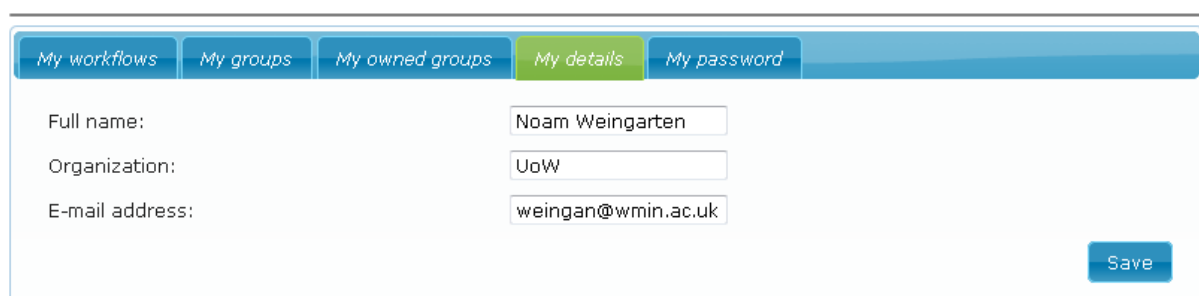



Figure 7: My Details tab

Home



My workflows My groups My owned groups My details My password

Password:

Save

Figure 8: My Password tab

4. WORKFLOW MANAGEMENT

4.1 LIST AND SEARCH VALIDATED WORKFLOWS

4.1.1 WORKFLOW BROWSE VIEW

Figure 9 below, displays the Workflow Browse view. This view displays most of the information about validated workflows and validated implementations in a convenient form.

Inputs, Outputs and Datasets can be expanded, to display more information.

Find Workflows

The screenshot shows the 'Find Workflows' interface. At the top, there is a search bar with a dropdown menu set to 'All Domains', a 'Fetch' button, and buttons for 'Search', 'Show All', and 'Refresh'. Below this is a pagination bar showing '(1 of 1)' and a list of workflow cards. The first card is for 'Workflow: FetchImages'. It has a 'Details' button in the top right. The card is divided into two main sections: 'Workflow Summary' and 'Implementation Preview (1)'. The 'Workflow Summary' section contains the following information: Domain: Demonstration, Application: Shiwa Image Manipulation Demo, Owner: Tamas Kukla, Group: shiwaExampleWfs, Status: validated, Keywords: web service, images, and Description: This workflow fetches images using a web service and puts them in a zip file. Used for demonstration purposes. Below this summary are three expandable sections: 'Inputs (1)', 'Outputs (1)', and 'Data sets (1)', each with a plus icon. The 'Implementation Preview (1)' section shows a diagram of the workflow and details for 'FetchImagesTaverna1.7', including Engine: Taverna(1.7), Version: 1.0, DCIs: SHIWA VO, Keywords: Taverna, Images, Web Service, and Description: Taverna 1.7 implementation of the workflow that is designed... At the bottom of the screenshot is another pagination bar showing '(1 of 1)' and a list of workflow cards.

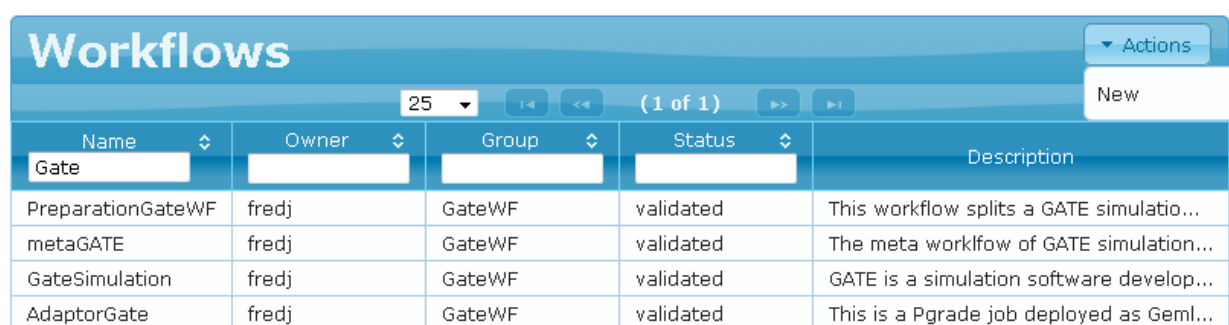
Figure 9: Workflows Browse view

Textual searches in workflow or implementation records can be performed using the search box. These searches can be restricted to domains of research.

By clicking the Details button, more information is displayed about the Workflow and its validated implementations.

4.1.2 WORKFLOW TABLE VIEW (FIGURE 10)

The Workflows Table view can be used to list and filter validated workflows by Name, Owner or Group.



Name	Owner	Group	Status	Description
Gate				
PreparationGateWF	fredj	GateWF	validated	This workflow splits a GATE simulation...
metaGATE	fredj	GateWF	validated	The meta workflow of GATE simulation...
GateSimulation	fredj	GateWF	validated	GATE is a simulation software develop...
AdaptorGate	fredj	GateWF	validated	This is a Pgrade job deployed as Geml...

Figure 10: Workflows table view

All available information about workflows and their validated implementations can be viewed by selecting the workflow from this view (see sections 4.1.3 Workflow details (Figure 11), 4.1.4 Workflow Attributes (Figure 12), 4.1.5 Workflow files & Download (Figure 13) and 4.1.6 Workflow Implementations (Figure 14))

The Action->New can be used to create new workflows (see 4.2 Create Workflows)

4.1.3 WORKFLOW DETAILS (FIGURE 11)

Selected workflow: FetchImages



Details	Owner	Access	Attributes	Files	Implementations
ID: 1654					
Description: This workflow fetches images using a web service and puts them in a zip file. Used for demonstration purposes.					

Figure 11: Workflow details page

4.1.4 WORKFLOW ATTRIBUTES (FIGURE 12)

Selected workflow: FetchImages

Details	Owner	Access	Attributes	Files	Implementations
Attributes					<div>▼ Actions</div> <div>Expand/Collapse</div> <div>Reload</div>
Name	Value				
▼ inputs					
▼ port0001					
datatype	file				
description	This file contains the data request for fetching images from a given web service.				
title	Input zip				
▼ outputs					
▼ port0002					
datatype	file				
description	This file contains the gathered images.				
title	Output zip				
▼ datasets					
▼ dataset0001					
description	An example dataset providing an input				
port0001	example_input.zip				Download
tasktype	demonstration				
application	Shiwa Image Manipulation Demo				
domain	Demonstration				
keywords	web service, images				

Figure 12: Workflow attributes page

The Expand/Collapse button in the Action control can be used to display all the attributes of the workflow.

Workflow attributes can be listed by clicking on the Attributes tab of a particular Workflow.

Table 1 describes the metadata structure of the attributes, and provides example values Figure 57 presents the Workflow metadata structure. These attributes allow straightforward categorisation of workflows and improve the browsing and search operations significantly. The input and output attributes with their sub-attributes define inputs and outputs of the Workflow. The dataset attribute specifies values of input parameters passed to workflow inputs, and they can also specify example outputs.

SHIWA Workflow Repository – Administrator Manual

Workflow metadata			Example value	Description	Table	Type	Mapping to SHIWA Desktop
id			1001	workflow identifier	workflow	int	
name			Factorial	workflow name	workflow	string	
owner_id			1008	workflow owner id	workflow	int	
group_id			exampleGroup	user group for defining access rights	workflow	int	
group_read			TRUE	whether group members can see wf/impl. data	workflow	bool	
group_download			TRUE	whether group members can download wf/impl. files	workflow	bool	
group_modify			TRUE	whether group members can modify wf/impl. data and upload files	workflow	bool	
others_read			TRUE	whether registered users can see wf/impl. data	workflow	bool	
others_download			TRUE	whether registered users can download wf/impl. files	workflow	bool	
published			TRUE	whether unregistered users can see wf/impl. data and download files	workflow	bool	
created			6/5/2011 13:12	workflow creation time	workflow	timestamp	
modified			6/7/2011 16:59	time of last modification	workflow	timestamp	
application			GATE	name of the application which the wf is part of	workflow_attr.	string	workflow->shiwa:application
description			This workflow ...	workflow description	workflow_attr.	string	
domain			Mathematics	scientific domain	workflow_attr.	string	workflow->shiwa:domain
keywords			factorial, integer	workflow keywords	workflow_attr.	string	
tasktype			demo	type of task the workflow represents	workflow_attr.	string	workflow->shiwa:tasktype
inputs	\			list of workflow inputs			
		port0001		first input port			shiwa:inport
		title	PositiveInteger	name of the port	workflow_attr.	string	shiwa:inport->dc:title
		datatype	file	data type of port	workflow_attr.	string	shiwa:inport->rdf:datatype
		description	this file contains an integer	port description	workflow_attr.	string	shiwa:inport->dc:description
outputs	\			list of output ports			
		file0002		first output port			shiwa-outport
		title	Factorial	name of the port	workflow_attr.	string	shiwa:outport->dc:title
		datatype	file	data type of port	workflow_attr.	string	shiwa:outport->rdf:datatype
		description	this file contains the factorial of the input integer	port description	workflow_attr.	string	shiwa:outport->dc:description
datasets	\			List of input/output configurations			
		dataset0001		First configuration			
		description	This dataset...	An example dataset	workflow_attr.	string	shiwa:dataset->dc:description
		port0001	input.dat	example value for port0001	workflow_attr.	string	shiwa:portref->rdf:value
		port0002	output.dat	example value for port0002	workflow_attr.	string	shiwa:portref->rdf:value

Table 1: Workflow metadata attributes

4.1.5 WORKFLOW FILES & DOWNLOAD (FIGURE 13)

Selected workflow: FetchImages

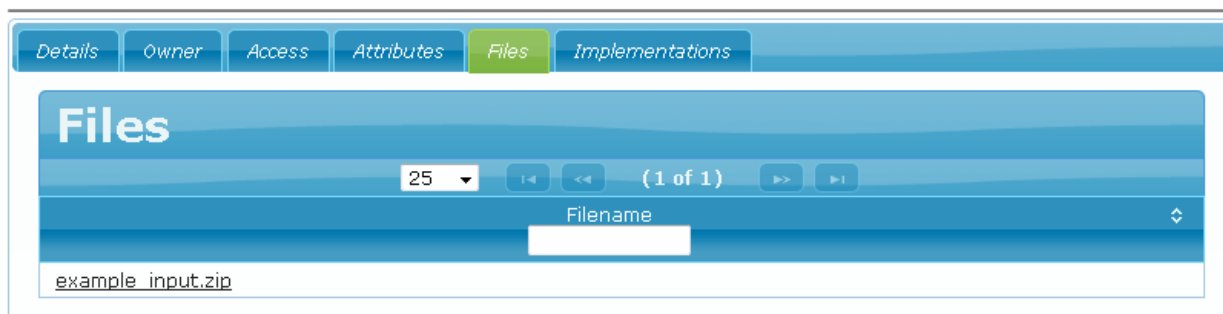


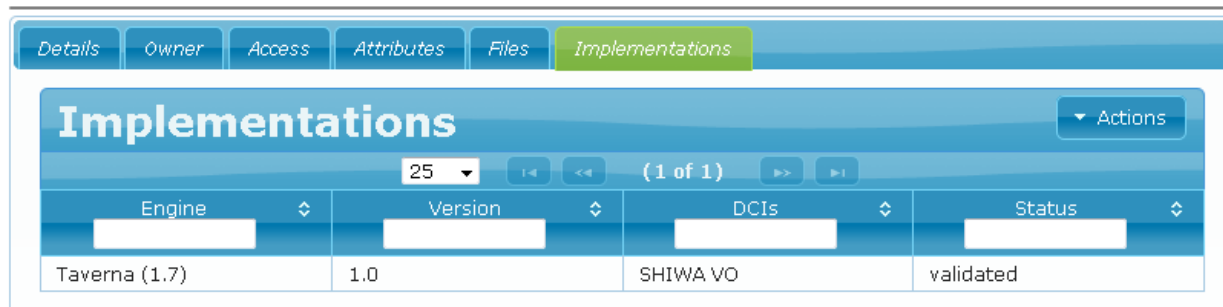
Figure 13: Workflows files page

Files associated with the abstract workflow definition can be downloaded from this page by right-clicking the files and selecting “Save as” or appropriate, as per your browser.

If files are not permitted to be downloaded, only their names will be displayed, but no live-links for download will be generated.

4.1.6 WORKFLOW IMPLEMENTATIONS (FIGURE 14)

Selected workflow: FetchImages



Engine	Version	DCIs	Status
Taverna (1.7)	1.0	SHIWA VO	validated

Figure 14: Workflows Implementations page

This lists the implementations of the selected workflow.

Selecting any of the validated implementations will direct you to the Implementation details of that Implementation (see 5.1.3 Implementations details (Figure 25)).

4.2 CREATE WORKFLOWS

Workflows can be created by selecting the *Create workflow* option from the toolbar (see Figure 15) and the New Workflow interface should be used to create the workflow (see Figure 16).

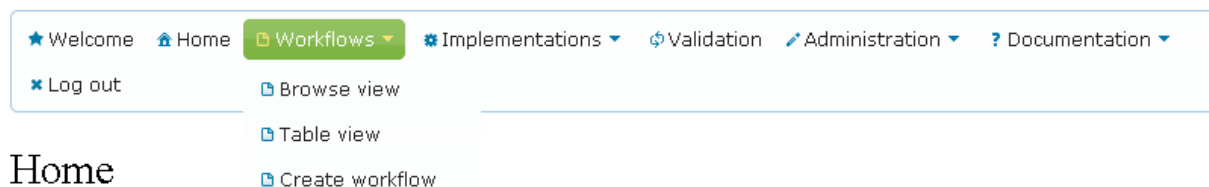


Figure 15: Create workflow tool

New workflow



Figure 16: New Workflow interface

Note: Workflows may only be associated with groups of which you are a member.

SHIWA Workflow Repository – Administrator Manual

The Repository administrator should be contacted, to create you a group, or put you in contact with a group owner.

The Workflow, now created, can be modified (see below).

4.3 MODIFY/DELETE WORKFLOW

On selecting an owned or modifiable workflow from the Table view of workflows (see Figure 10) or from your My Workflows view (see Figure 6), the workflow may be modified.

Selected workflow: SimpleWF_IntegerSubtractor

The screenshot shows the 'Workflow Details' tab for the workflow 'SimpleWF_IntegerSubtractor'. The interface includes a top navigation bar with tabs: 'Details' (selected), 'Owner', 'Access', 'Attributes', 'Files', and 'Implementations'. On the right, there is an 'Actions' dropdown menu with a 'Delete' option. The main content area displays the workflow's ID as '1601' and a description: 'This workflow subtracts two integers and outputs the result. The input integers are provided in text files and the result is also a text file containing the difference. This workflow serves demonstration purposes.' A 'Save' button is located at the bottom right of the description area.

Figure 17: Workflow details tab

The Workflow Details tab can be used to modify the workflow description.

Note: the Action button can be used to delete the workflow.

4.3.1 OWNERSHIP

Selected workflow: manual

The screenshot shows the 'Workflow Owner' tab for the workflow 'manual'. The interface includes a top navigation bar with tabs: 'Details', 'Owner' (selected), 'Access', 'Attributes', 'Files', and 'Implementations'. On the right, there is an 'Actions' dropdown menu. The main content area displays the 'Owner' field with the value 'noam' entered in a text input box. A 'Save' button is located at the bottom right of the input area.

Figure 18: Workflow Owner tab

The Workflow Owner tab can be used to change ownership of the workflow

4.3.2 ACCESS CONTROL

Selected workflow: SimpleWF_IntegerSubtractor

▼ Actions

DetailsOwnerAccessAttributesFilesImplementations

Access rights

Group name:

shiwaExampleWfs

	Read	Download	Modify
Group:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Others:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Note that if you enable others to read/download your workflow, it will be accessible to registered users only. A validator will look at your workflow to make sure that the provided data is correct. Once the validator accepts your workflow, it's status will change to validated and it will be publicly available to guest users as well. Once a workflow has been validated, it cannot be modified.

Validation

Validated:

☐

Save

Figure 19: Workflow Access Control tab

The Workflow owner or an Administrator may modify the other access controls as per requirements.

Only Validators may validate workflows, by ticking the validation tick box.

4.3.3 ATTRIBUTES

Selected workflow: MetaWF_ImageManipulationDemo

▼ Actions

DetailsOwnerAccessAttributesFilesImplementations

Attributes

Name	Value	Actions
▼ inputs		Add
▼ port0001		Remove
datatype	file	Edit
description	This file contains the data request for fetching images from a given web service.	Edit
title	Input zip	Edit
▼ outputs		Add
▼ port0002		Remove
datatype	file	Edit
description	This file contains the manipulated images.	Edit
title	Output zip	Edit
▼ datasets		Add
▼ dataset0001		Add Remove
description	An example dataset providing an input	Edit
port0001	example_input.zip	Download Edit Remove
tasktype	demonstration	Edit
application	Shiwa Image Manipulation Demo	Edit
domain	Demonstration	Edit
keywords	web service, images, edge highlighting, RGB	Edit

▼ Actions
Expand/Collapse
Reload
Save

Figure 20: Workflow Attributes tab

Workflow attributes can be specified using this interface, see Table 1 for the Workflow metadata attributes.

4.3.4 FILES


Selected workflow: MetaWF_ImageManipulationDemo

▼ Actions

DetailsOwnerAccessAttributesFilesImplementations

Files

25 (1 of 1)

Filename
 example_input.zip

▼ Actions
Upload
Delete

Figure 21: Workflow Files tab

Files can be uploaded to or deleted from the Workflow using the above interface.

4.3.5 IMPLEMENTATIONS

Selected workflow: MetaWF_ImageManipulationDemo

▼ Actions

DetailsOwnerAccessAttributesFilesImplementations

Implementations

25<1<<<(1 of 1)>>>>1

EngineVersionDCIsStatus

P-GRADE (2.4.1)1.0SHIWA VOvalidated

▼ ActionsNew

Figure 22: Workflows Implementations tab

Implementations of workflows can be created using this tab, as will be explained below.

5. IMPLEMENTATION MANAGEMENT

5.1 LIST AND SEARCH IMPLEMENTATIONS

5.1.1 IMPLEMENTATION BROWSE VIEW

Figure 23 below, displays the Implementation Browse view. This view displays most of the information about validated implementations in a convenient form.

Dependencies and Configurations can be expanded, to display more information.

Find Implementations

All Domains

(1 of 1)

Workflow: FetchImages | Engine: Taverna(1.7) | Implementation version: 1.0

Graph



Implementation summary

Title: FetchImagesTaverna1.7
Workflow: FetchImages
Engine: Taverna(1.7)
Version: 1.0
Status: validated
Validator: Tamas Kukla
Language: Scufi
Domain: Demonstration
DCIs: SHIWA VO
GEMMLCA id: Taverna-WF2g
Licence: Creative Commons Attribution 3.0 Unported License
Definition: [fetchImages.xml](#)
Keywords: Taverna, Images, Web Service

Description: Taverna 1.7 implementation of the workflow that is designed to be executed on ngs worker nodes.

Dependencies (3)

Name	Type	Description
ImageWebService	Service	A web service for gathering images.
Parameter Mapping	Other	This file maps files of the input zip to workflow ports.
VO for execution in SSP	DCI	The VO the user has to be member of to execute this workflow in the SSP

Configurations (1)

Configuration 1

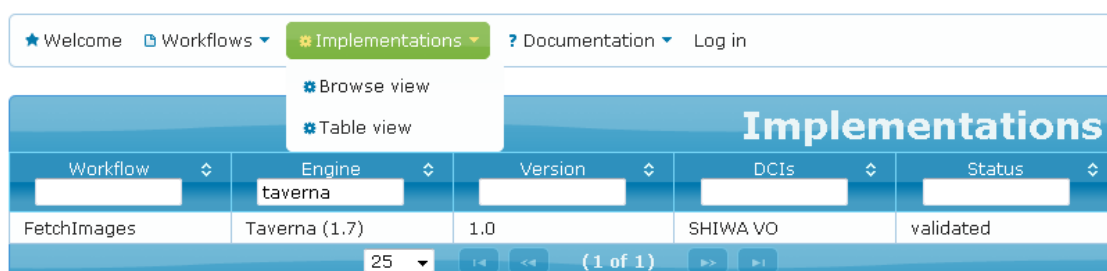
Dependency Name	Value
ImageWebService	http://moby.ucalgary.ca/moby/MOBY-Central.pl
Parameter Mapping	example_params.map
VO for execution in SSP	SHIWA VO

Figure 23: Browse Implementation view

Textual searches in workflow records can be performed using the search box. These searches can be restricted to specific domains of research.

5.1.2 IMPLEMENTATION TABLE VIEW

The Implementations Table View can be used to list and filter validated Implementations by Workflow, Engine, Version or DCI.



Workflow	Engine	Version	DCIs	Status
FetchImages	Taverna (1.7)	1.0	SHIWA VO	validated

Figure 24: Implementations Table view

All available information about Implementations can be viewed by selecting the Implementation from this view (see sections 5.1.3 Implementations details (Figure 25), 5.1.4 Implementation Attributes and 5.1.5 Implementation files & Download)

5.1.3 IMPLEMENTATIONS DETAILS (FIGURE 25)

Workflow: FetchImages
Engine: Taverna(1.7)
Implementation version: 1.0

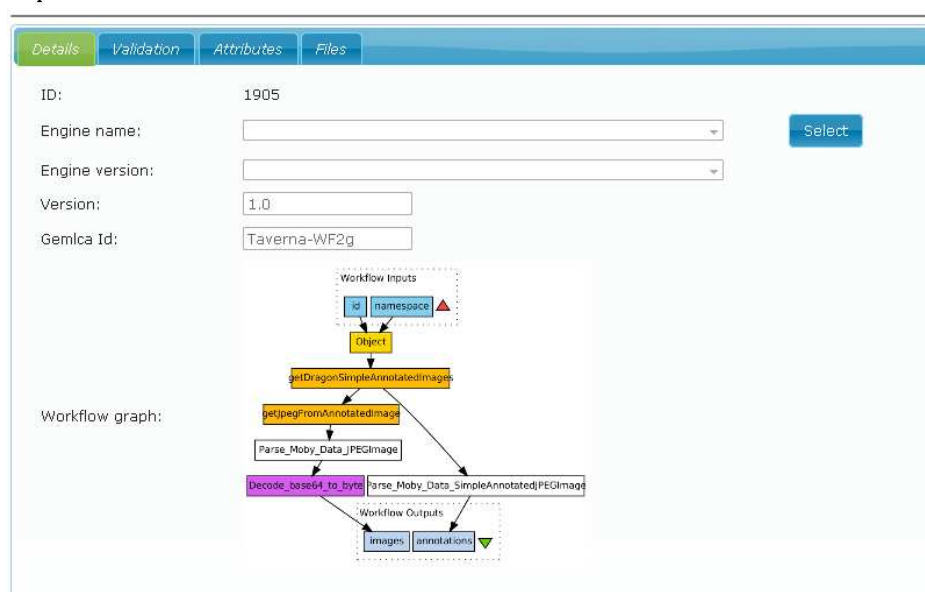


Figure 25: Implementation Details page

The *Validation* tab can be used to see the validation life-cycle status.

The *Attributes* tab can be used to open the attributes of the selected Implementation.

The *Files* tab can be used to list and download files held in the repository for the selected implementation.

5.1.4 IMPLEMENTATION ATTRIBUTES

Implementation attributes can be opened by clicking on the attributes tab of a given implementation as illustrated in Figure 26. The left column of the attributes table contains attribute names, while the right column contains attribute values.


Similarly to workflows, the metadata template is used to help the definition of most common attributes. The three key attributes are: definition, dependencies and configurations. The definition attribute, is the workflow definition file i.e. the executable to be interpreted by the workflow engine. The dependency attribute can be any requirement of the particular implementation. These can include for instance files, executables, libraries or VO memberships required for execution. Configuration attributes resolve these dependencies.

Table 2 describes each attribute and provides example values. Figure 57 illustrates the metadata structure.

Implementation metadata	Example value	Description	Table	Type	Mapping to SHIWA Desktop
id	1002	implementation identifier	implementation	int	
workflow_id	1001	identifier of the abstract workflow that the impl. implements	implementation	int	
engine_id	1005	workflow engine identifier	implementation	int	
version	1.01	implementation version	implementation	string	
created	6/15/2011 4:12	workflow creation time	implementation	timestamp	workflow->dcterms:created
modified	6/21/2011 11:24	time of last modification	implementation	timestamp	workflow->dcterms:modified
state	VALIDATED	implementation status	implementation	enum	
uuid	1234-1234-1234	uuid of implementation	imp_attr.	string	workflow->dc:identification
title	FetchImagesTaverna	title of the implementation	imp_attr.	string	workflow->dc:title
description	this implementation is	implementation description	imp_attr.	string	workflow->dc:description
definition	workflow.xml	workflow descriptor file	imp_attr.	string	workflow->shiwa:definition
graph	workflow.png	workflow graph screenshot	imp_attr.	string	
language	SCUFL	language of the workflow descriptor	imp_attr.	string	workflow->shiwa:language
rights	© SHIWA	copyright information	imp_attr.	string	workflow->dc:rights
licence	Demo licence	licence information	imp_attr.	string	workflow->dcterms:licence
keywords	Taverna, Images, Web Service	keywords used for searching	imp_attr.	string	
dependencies	\	List of dependencies: files needed for executing factorial.sh. It can be empty in the case of DGs.	imp_attr.		
	dep0001 \	first dependency	imp_attr.		shiwa:dependency
	title	Image Service	imp_attr.	string	shiwa:dependency->dc:title
	description	A web service for gathering images.	imp_attr.	string	shiwa:dependency->dc:description
	dep0002 \	second dependency	imp_attr.		shiwa:dependency
	title	Parameter Mapping	imp_attr.	string	shiwa:dependency->dc:title
	description	This file maps files of the input zip to workflow ports.	imp_attr.	string	shiwa:dependency->dc:description
configurations	\	List of dependency configurations. A configuration resolves all dependencies of the executable. It can be empty if no dependencies.	imp_attr.		
	conf0001 \	first configuration	imp_attr.		
	description	This configuration...	imp_attr.	string	shiwa:configuration->dc:description
	dep0001	http://moby.ucalgary.ca/...	imp_attr.	string	shiwa:dependencyref->rdf:value
	dep0002	example_params.map	imp_attr.	string	shiwa:dependencyref->rdf:value

Table 2: Implementation metadata attributes

Workflow: FetchImages
Engine: Taverna(1.7)
Implementation version: 1.0



Name	Value
▼ dependencies	
▼ dep0001	
type	Service
description	A web service for gathering images.
title	ImageWebService
▼ dep0002	
type	Other
description	This file maps files of the input zip to workflow ports.
title	Parameter Mapping
▼ dep0003	
type	DCI
description	The VO the user has to be member of to execute this workflow in the SSP
title	VO for execution in SSP
▼ configurations	
▼ conf0001	
dep0001	http://moby.ucalgary.ca/moby/MOBY-Central.pl
dep0002	example_params.map
dep0003	SHIWA VO
title	FetchImagesTaverna1.7
description	Taverna 1.7 implementation of the workflow that is designed to be executed on ngs worker nodes.
definition	fetchImages.xml
graph	Taverna_wf2.png
language	Scufl
rights	
licence	Creative Commons Attribution 3.0 Unported License
keywords	Taverna, Images, Web Service
uuid	

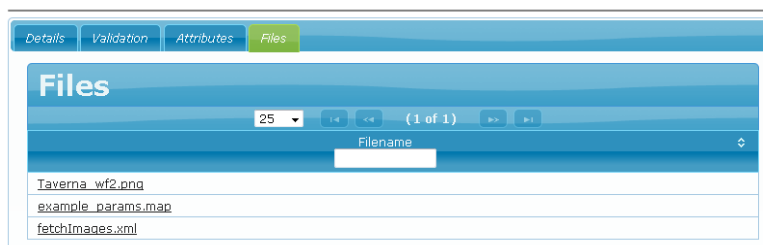
Figure 26: Implementation attribute table

The definition file can be downloaded from this page by clicking on the *download* live-link.

5.1.5 IMPLEMENTATION FILES & DOWNLOAD

As with Workflow files, these can be downloaded from the Implementation Files view, by clicking on the respective live-link (see Figure 27).

Workflow: FetchImages
Engine: Taverna(1.7)
Implementation version: 1.0



Filename	Download
Taverna_wf2.png	
example_params.map	
fetchImages.xml	

Figure 27: Implementation Files and download

5.2 CREATE IMPLEMENTATION

See 4.3.5 Implementations

5.3 MODIFY IMPLEMENTATION

5.3.1 DETAILS

Workflow: SimpleWF_IntegerSubtractor

Engine: Kepler(1.0)

Implementation version: 1.1

Actions

DetailsValidationAttributesFiles

ID:2203

Engine name:Kepler

Select

Engine version:1.0

Version:1.1

Gemlca Id:Kepler-Subtract

Workflow graph:

Expression Reader2

SDF Director

Add or Subtract2

Expression Reader

File Writer

Save

Figure 28: Modify Implementation details

The above interface can be used to modify the details of implementations the user has permissions to modify.

The Actions button can be used to change the life-cycle status, and may be used to delete the implementation.

5.3.2 ATTRIBUTES

Workflow: SimpleWF_IntegerSubtractor

Engine: Kepler(1.0)

Implementation version: 1.1

▼ Actions

DetailsValidationAttributesFiles

Attributes

▼ Actions

Name	Value	
▼ dependencies		Expand/Collapse
▼ dep0001		Add Reload
type	DCI	Remove Save
description	The VO the user has to be member of to execute this workflow in the SSP.	Edit
title	VO for execution in SSP	Edit
▼ configurations		Add
▼ conf0001		Add Remove
dep0001	SHIWA VO	Edit Remove
title	Kepler Subtract Demo	Edit
description	This workflow is executed locally to the Kepler engine.	Edit
definition	Kepler_wf1.xml	Download Edit
graph	Kepler_wf1.png	Download Edit
language	MOML	Edit
rights	SHIWA project	Edit
licence	Apache License, Version 2.0	Edit
keywords	Kepler, local, subtract, integer	Edit
uuid		Edit

Figure 29: Modify implementation attributes

This interface can be used to modify the implementation attributes. See Table 2 for the metadata structure.

The Expand/Collapse action can be used to expand or collapse the listing.

5.3.3 FILES

Workflow: SimpleWF_IntegerSubtractor
Engine: Kepler(1.0)
Implementation version: 1.1

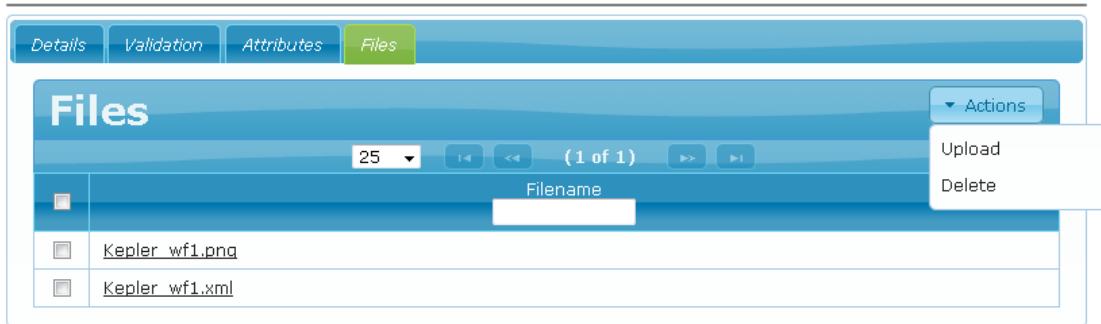


Figure 30: Modify implementations - Files tab

5.3.4 GRAPH IMAGES OF IMPLEMENTATIONS

A screenshot of the implementation can be made, from the native workflow graphical editor, which can then be posed in the repository with the implementation (see Figure 28). The image should either be a jpg, png or gif.

The resolution required can depend on the workflow complexity - note that the thumbnail images will have a width of 150 pixels, and so the resolution should be sufficient to provide a recognizable thumbnail at 150 pixels across.

The Image file should be uploaded to the implementation, as a file (see Figure 30)

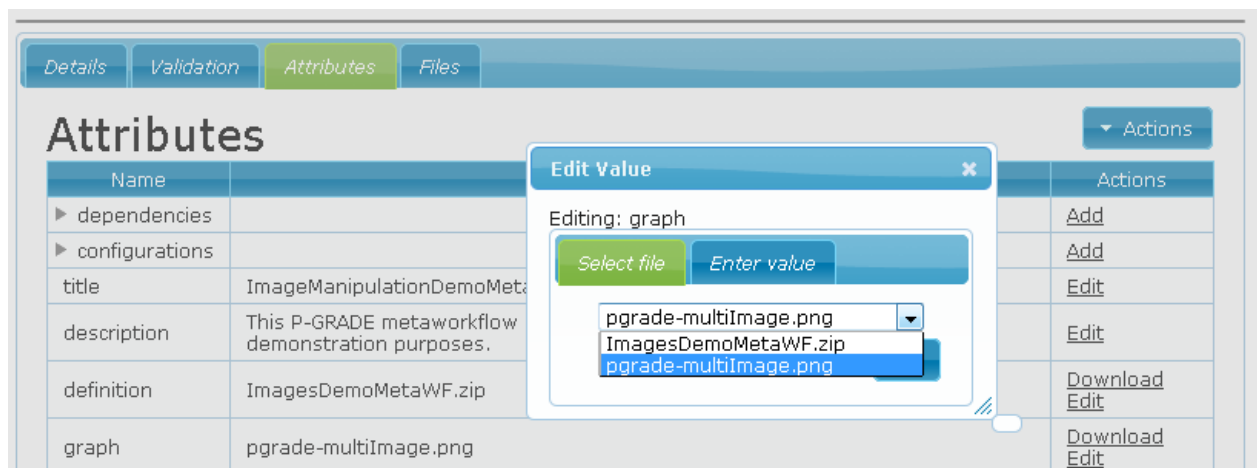


Figure 31: Selecting graph image file

Once uploaded, the file can be selected from the uploaded implementation files, on editing the graph attribute of the said implementation (see Figure 31).

5.3.5 VALIDATION

The Validation tab can be used to view the life-cycle status (see Figure 48). Only actions permitted to that user will be made available.

5.3.6 WORKFLOW EXECUTION

See the Workflow Execution Extension below, for information on how to deploy the Implementation to GEMICA, to enable workflow execution from the SSP.

6. IMPORT WORKFLOWS FROM MYEXPERIMENT

6.1 PREFACE

The SHIWA Repository is integrated with myExperiment which is a social networking site and Virtual Research Environment (VRE) designed for people to share, discover and reuse workflows.

The SHIWA Repository GUI is extended to facilitate browsing and importing publically shared Taverna1 workflows on the myExperiment site. This feature of the SHIWA Repository can be accessed using the main menu bar on the top. This feature of the GUI can only be viewed after logging in (see Figure 37).

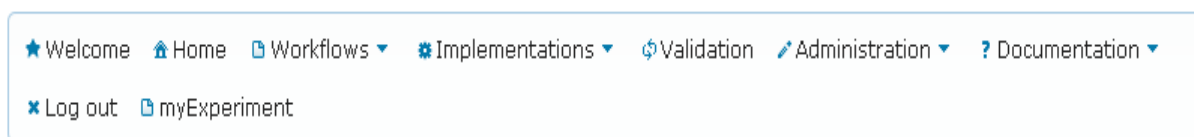


Figure 32: Accessing myExperiment integration feature

The following chapters go through importing workflows from myExperiment to the SHIWA Repository and describe the provided functionality.

6.2 IMPORTING WORKFLOWS

All available information about finding and importing workflows from myExperiment to the SHIWA Repository can be viewed by selecting “myExperiment” from the menu bar on the top. The provided link (<http://www.myExperiment.org>) on the page can be used to browse publicly shared workflows on the myExperiment site.

A workflow can be downloaded from the myExperiment site to the SHIWA Repository by entering the ID of the workflow (ID of a workflow on the myExperiment site can be found from its URL, for example the ID of a workflow with the URL; <http://www.myExperiment.org/workflows/90.html> is 90) into the provided box and click the “Import” button (see Figure 33).



Import workflows from myExperiment Repository

You can use <http://www.myExperiment.org> to find publicly shared workflows on the myExperiment workflow repository. If you want to download a workflow from the myExperiment repository to the SHIWA repository, please enter ID of the workflow (ID of a workflow on the myExperiment can be found from its URL, for example the ID of a workflow with the URL; <http://www.myexperiment.org/workflows/90.html> is 90) into the box below and click the Import button.

Import workflow

Workflow ID:

Import

Figure 32: myExperiment workflow import page

By entering the ID a workflow into the provided box and clicking the “Import” action button, the workflow can be imported from myExperiment site and a new workflow and implementation can be created automatically in the SHIWA Repository. Additionally the required workflow files are uploaded; messages are displayed on the top of the page about the success or failure of the operation (see Figure 39).

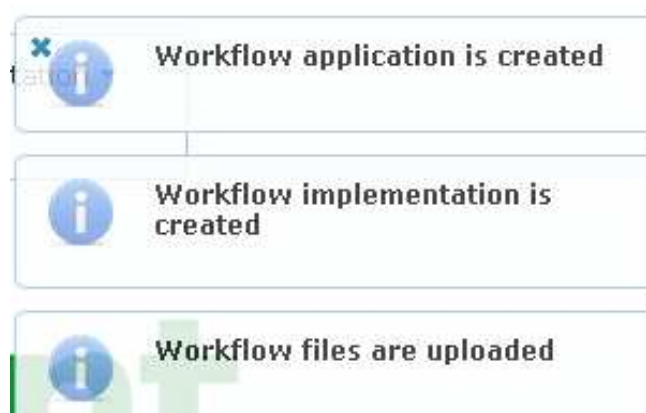


Figure 33: Importing a workflow from myExperiment

The newly created workflows can be found using Workflows Table or Workflows Browse view. The workflow names are changed slightly by removing empty spaces (see Figure):

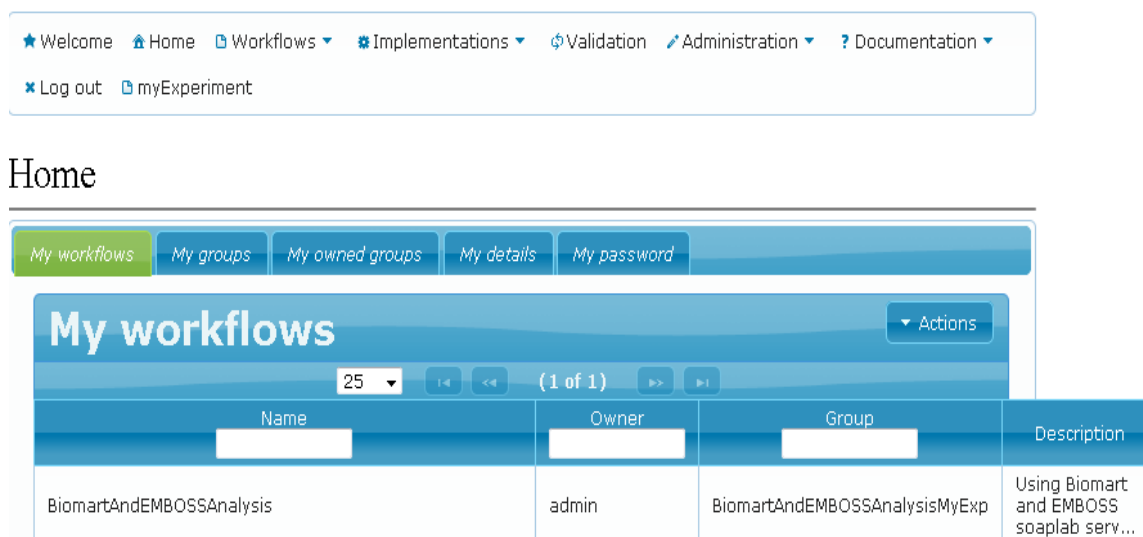


Figure 35: Newly created workflow

6.3 INPUT PORTS CONFIGURATION

If a workflow has input ports, the input data should be put in files and uploaded to the workflow application. The correspondent input ports are configured using the uploaded files (see Figure 36).

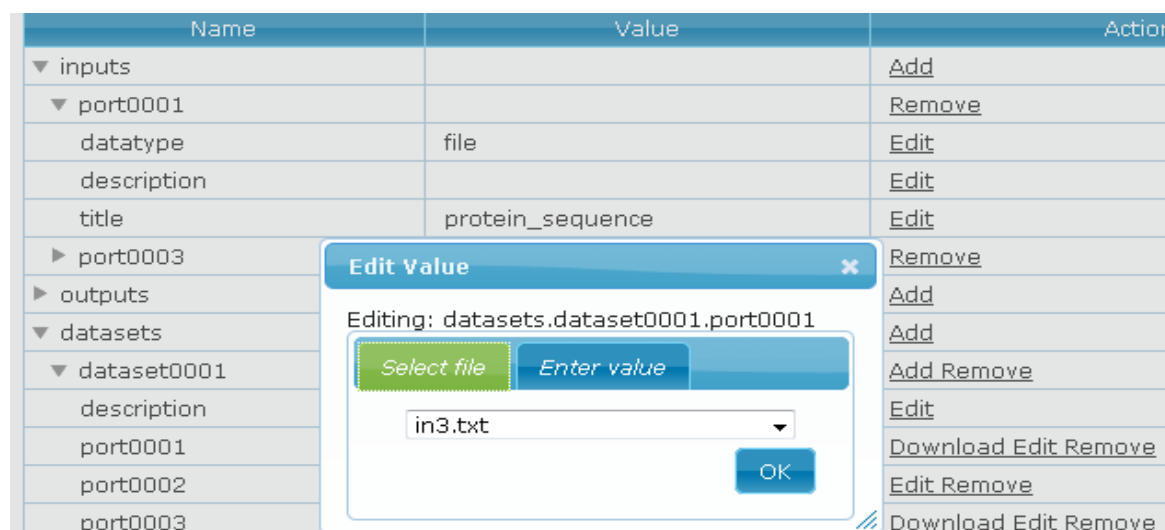


Figure 36: Configuring input ports

The following chapter outline the deployment of the workflows to the GEMLCA service in order to run a workflow implementation in the SSP.

7. WORKFLOW EXECUTION EXTENSION

7.1 PREFACE

In order to run a workflow implementation in the SSP, it must be deployed to the GEMLCA Service. To deploy an implementation the executable should be transferred to the GEMLCA Service, and the execution parameters should be described to enable the execution of the workflow implementation using the SHIWA Portal.

The List of Workflow Implementations (see Figure 37) displays the implementation's execution status, i.e. the workflow implementation has been already deployed and can be executed on the simulation platform. Deployed workflows have the blue icon in the left column.

Selected workflow: SimpleWF_IntegerSubtractor



Run	Engine	Version	DCIs	Status
	Kepler (1.0)	1.3	SHIWA VO	validated
	Kepler (1.0)	1.2	SHIWA VO	validated
	Kepler (1.0)	gemlcaTest	SHIWA VO	new
	Kepler (1.0)	screenshots	SHIWA VO	new
	Kepler (1.0)	1.2_noam2	SHIWA VO	new
	Kepler (1.0)	1.2_noam	SHIWA VO	new
	Kepler (1.0)	1.1	SHIWA VO	new

Figure 37: Implementations list

7.2 CREATE EXECUTION

Once the execution of a workflow implementation has been correctly and completely described in the repository, it is ready to be deployed to the GEMLCA Service, and made available for execution through the SHIWA Simulation Portal. The workflow implementation's execution can be created and configured from the Implementation's Attributes page (see Figure 38).

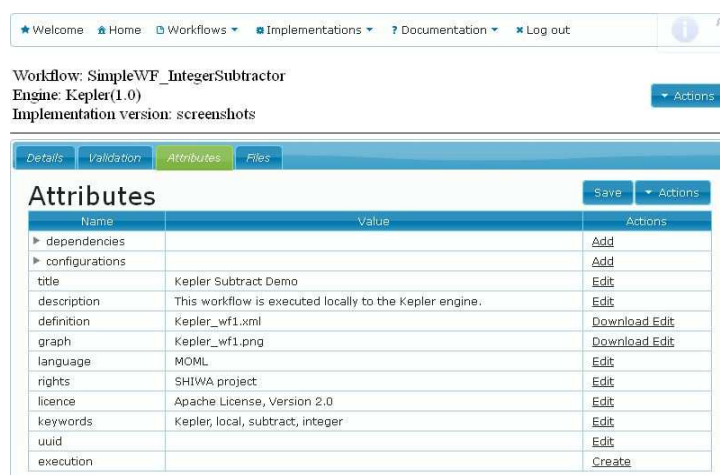


Figure 38: Implementation Attributes

On clicking the “Create” action tab of the execution attribute, a dialog box is displayed. The workflow engine which interprets and executes this implementation should be selected from the list (see Figure 39).

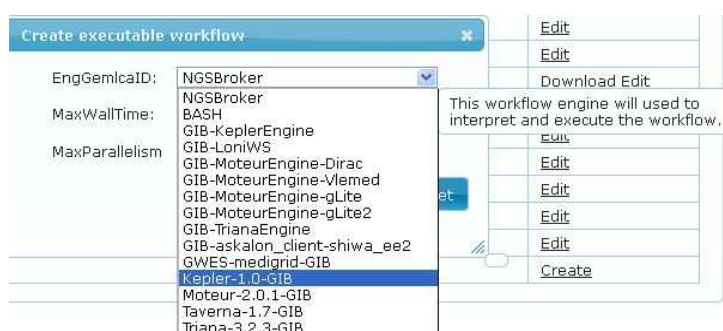


Figure 39: Select workflow engine interpreter

Advanced users can configure details of the execution back-end at this stage (see Figure 40):

- **Maximum Walltime:** This is the maximum execution time in minutes – after which the execution will be suspended even if it is still running.
- **Maximum Parallelism :** This is the maximum number of parallel jobs of a process

The "Create" button will create the workflow implementation's execution.

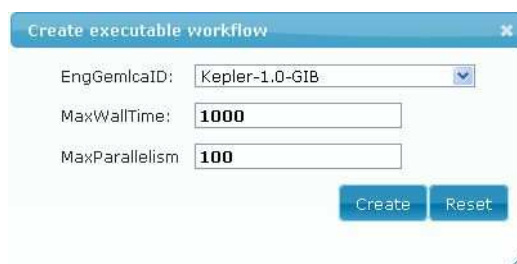


Figure 40: Configure Execution Back-end

7.3 CONFIGURE PARAMETERS

▼ execution		Remove
wfGemlcaId	SimpleWF_IntegerSubtractor-Kepler-1.0-screenshots	
engGemlcaId	Kepler-1.0-GIB	Select
maxWallTime	1000	Edit
maxParallelism	100	Edit
parameters		Add

Figure 41 : Configured Execution Back-end

The execution section of the Implementation's Attributes can now be expanded to add and define parameters by selecting the "Add" tab (see Figure 40).

First, the user should first drag down the ParameterID dialog, to select a unique parameter number. Next, the user should select the type of parameter they are describing.

Execution parameters can be categorized into 4 types (see Figure 41):

- **INPUT_PORT** and **OUTPUT_PORT** types of parameters should be used to configure inputs or outputs which have been described in the Workflow's Attributes (see 4.1.4 Workflow Attributes (Figure 12)).
- **DEPENDENCY** type parameters can be used to configure dependencies which have been described in the Implementation's Attributes (see 5.1.4 Implementation Attributes).
- **CUSTOM** type parameters can be used to configure parameters which are not described elsewhere in the repository, but are required for execution. These should only be used by advanced users.

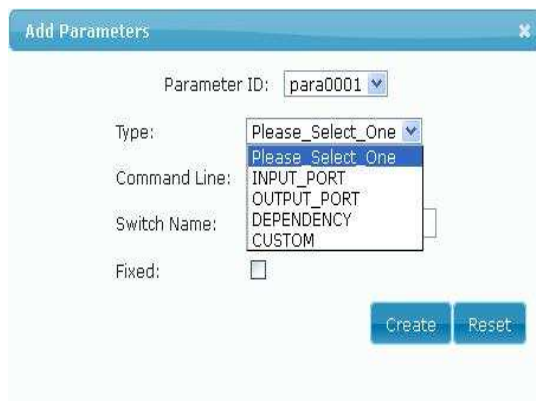


Figure 41 : Select Parameter Type


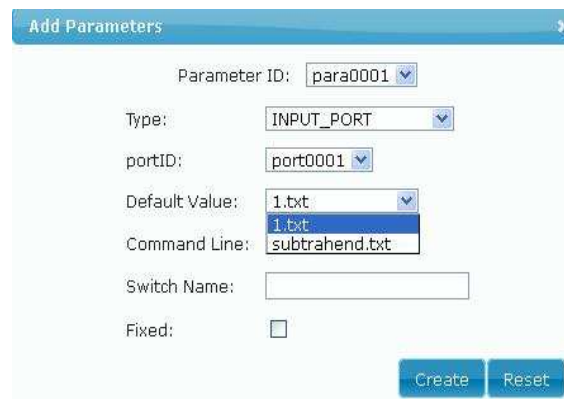


Figure 42 : Select port

If the parameter is a file it should be associated with a port. First, the user should select the port type and define its ID (see Figure 42) - as was configured in the Workflow's Attributes.

Next, the user should select which file to associate with the port as the default input file or output file (see Figure 43) in the "Default Value" area.



The screenshot shows the 'Add Parameters' dialog box with the following fields and values:

- Parameter ID: para0001
- Type: INPUT_PORT
- portID: port0001
- Default Value: 1.txt
- Command Line: 1.txt (highlighted), subtrahend.txt
- Switch Name: (empty)
- Fixed: ☐
- Buttons: Create, Reset


Figure 43 : Select file

Non-file type parameters, will display to the user in the dialog the ability to enter in the default value.

The user can also configure other details of this parameter:

- **Command Line:** This specifies whether this parameter is a command line parameter.
- **Switch Name:** If the parameter is a command line parameter it allows definition of the switch name.
- **Fixed:** This defines whether the user can modify this parameter from the SSP.
- **Title:** This is a brief description of the parameter of custom type parameters, equivalent to what would otherwise have been in the Workflow>Attributes>port>title field.

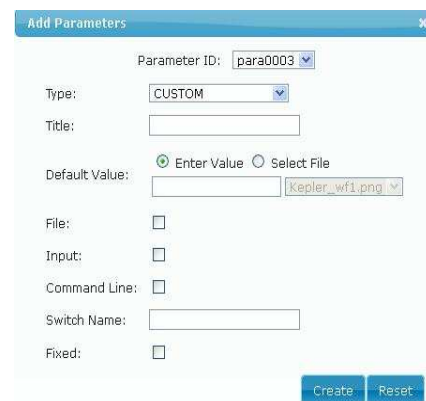
The user interfaces for DEPENDENCY and CUSTOM typed parameters are displayed in Figure 44 and Figure 45 respectively.



The screenshot shows the 'Add Parameters' dialog box with the following fields and values:

- Parameter ID: para0001
- Type: DEPENDENCY
- depID: dep0001
- Default Value: SHIWA VO
- Command Line: ☐
- Switch Name: (empty)
- Fixed: ☐
- Buttons: Create, Reset

Figure 44 : Dependency type parameter



The screenshot shows the 'Add Parameters' dialog box with the following fields and values:

- Parameter ID: para0003
- Type: CUSTOM
- Title: (empty)
- Default Value: Enter Value (selected), Select File (radio button), (empty text box), kepler_wf1.png
- File: ☐
- Input: ☐
- Command Line: ☐
- Switch Name: (empty)
- Fixed: ☐
- Buttons: Create, Reset

Figure 45 : Custom type parameter

Once the parameter is configured, the user should click "Create" button.

The user should repeat this process to describe all parameters.

7.4 DEPLOY IMPLEMENTATION TO THE GEMLCA SERVICE

Once the execution has been fully configured, the action tab displays the available actions (see Figure 46):

- **Expand/Collapse:** This can be used to either expand or collapse nested attributes.
- **Reload:** This will reload the last saved table of Implementation Attributes for this Implementation.
- **Save:** This will save the Implementation Attributes.
- **Deploy to GEMLCA:** This deploys the currently configured execution to GEMLCA, so that it can be executed from the SSP.
- **Undeploy from GEMLCA:** This un-deploys the currently deployed execution of this implementation from GEMLCA – thereby making no longer executable from the SSP.
- **Re-Deploy to GEMLCA:** This redeploys the currently deployed execution of this Implementation in GEMLCA, with the new configuration.

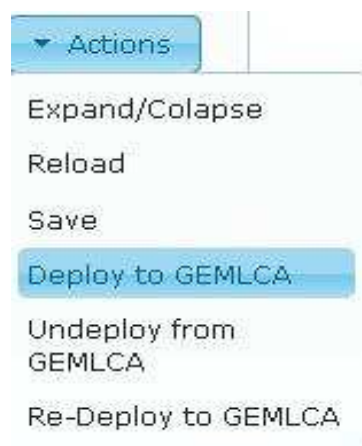


Figure 46 : Actions for deployment of execution

Not all of the actions in Figure 46 will be available, only the ones which are appropriate, given the state of execution deployment

Once deployed, the name of the GEMLCA Id will appear in the Implementation's Details page.

To display the deployment logs, please open Actions/Show Deployment Logs. Should the deployment fail, please copy the contents of the deployment logs, and email it to the Repository administrators, to help diagnose the issue.

At this stage, the workflow will be executable from the SSP.

7.6 MODIFY EXECUTION

Links to edit values in Figure 47 can be used to modify configurations of the execution. The revised execution can be saved and re-deployed using the appropriate action.

7.7 EXECUTION OF PROTOTYPE WORKFLOW FROM THE APPENDIX OF THE USAGE POLICY

Figure 47 should be considered to be appended to the end of Figure A.7.Simple

▼ execution		Remove
engGemlcaId	Kepler-1.0-GIB	Re-Select
maxWallTime	1000	Edit
maxParallelism	100	Edit
▼ parameters		Add
▼ para0001		Edit Remove
type	INPUT_PORT	
portId	port0001	
title	Subtrahend	
defaultValue	subtrahend.txt	
cmdLine	false	
switchName		
fixed	false	
file	true	
input	true	
▼ para0002		Edit Remove
type	INPUT_PORT	
portId	port0002	
title	Minuend	
defaultValue	minuend.txt	
cmdLine	false	
switchName		
fixed	false	
file	true	
input	true	
▼ para0003		Edit Remove
type	OUTPUT_PORT	
portId	port0003	
title	Difference	
defaultValue	difference.txt	
cmdLine	false	
switchName		
fixed	false	
file	true	
input	false	

Figure 47: Execution Related Metadata

8. IMPLEMENTATION LIFE CYCLE

Figure 48 below, shows the lifecycle of an implementation

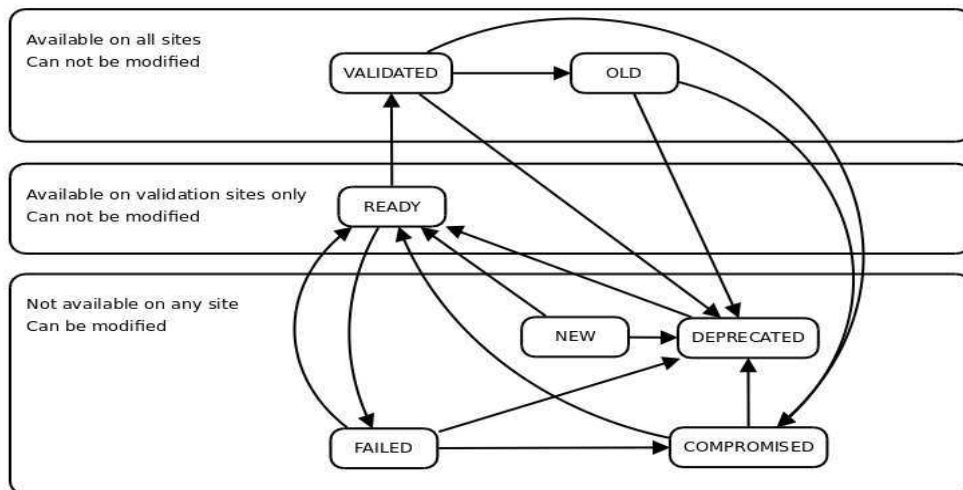


Figure 48: Implementation Life Cycle

9. WORKFLOW AND IMPLEMENTATION VALIDATION

See Figure 48 the Implementation life cycle.

9.1 OWNER

The Owner of a parent workflow of an implementation may make several changes to the lifecycle states of the implementation:

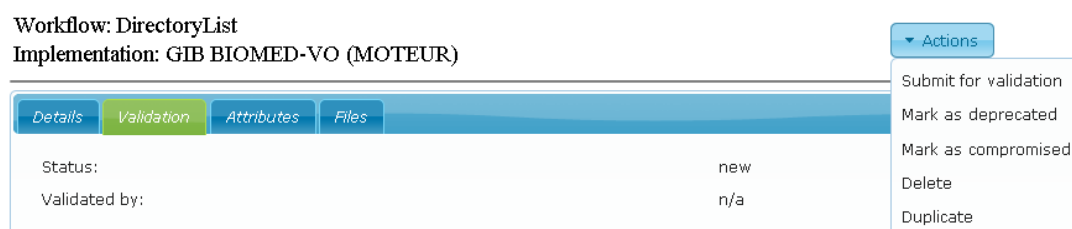


Figure 49: Implementation validation actions – Owner

- The owner may submit the implementation for validation by a validator.
- The owner may mark the implementation as being depreciated.
- The owner may mark the implementation as being compromised.
- The owner may delete the implementation.
- Finally, the owner may duplicate the implementation, so that the duplicated implementation can be developed forward as the next version.

Note: Once a user has submitted an implementation for validation, the validation may not be modified. Should the user wish to change the implementation, the recommendation is to duplicate the submitted implementation – creating a new implementation, which may be modified.

9.2 VALIDATOR

validator [active, validator]

★ Welcome Home Workflows Implementations Validation Documentation Log out

Validation and Publication

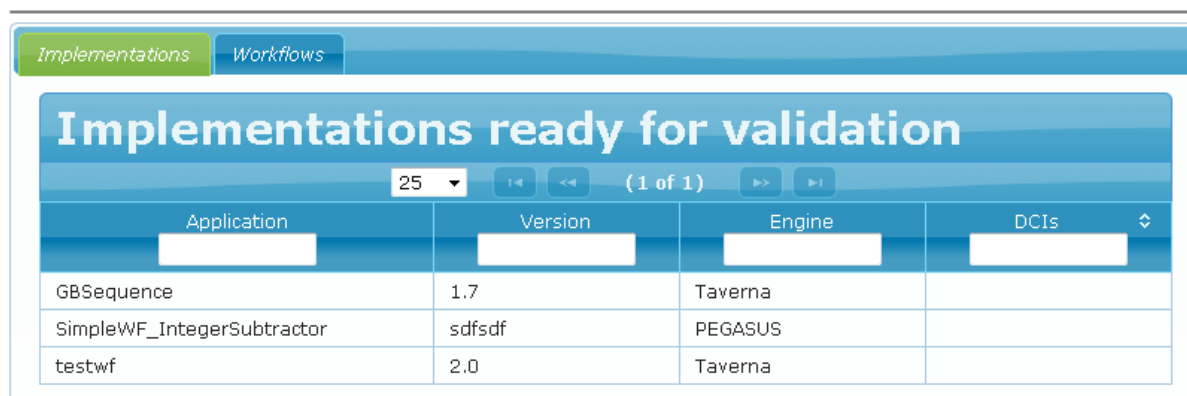


Figure 50: Validation interface

Only validators will have the *Validation* interface (see Figure 50).

The *Implementations* tab will list implementations which have been submitted for validation.

The *Workflows* tab will display workflows which are both visible to the validator and are yet to be validated.

On selecting an implementation, in the validation tab, the validator will be able to either accept or deny the validation, by clicking the Actions button.

On selecting workflows for validation, there will be a *Validation* tickbox available in the Access Control tab.

10. USER MANAGEMENT

Only administrators may manage users.

10.1 BROWSE USERS

Noam Weingarten [active, validator, admin]

★ Welcome 🏠 Home 📁 Workflows ⚙️ Implementations 🔍 Validation 🛠️ Administration ? Documentation

✕ Log out

🔗 Users
🔗 Groups
🔗 Engines

Users

25 (1 of 2)

Login name	Full name	Organization	Roles
admin	SHIWA Administrator	SHIWA	active, validator, admin
kuklat	Tamas Kukla	UoW	active
noam	Noam Weingarten	UoW	active, validator, admin
silvia	Silvia Delgado Olabarriaga	SHIWA	active
vladimir	Vladimir Korkhov	SHIWA	active

Figure 51: Browse users

Users can be listed, by selecting *Users* from the *Administration* tool.

10.2 CREATE USERS

The *Action* button from Figure 51 can be used to create new users.

10.3 MODIFY USERS

kuklat

Actions
Delete

Details Password Groups Owned groups

Full name: Tamas Kukla

Organization: UoW

E-mail address: kuklat@wmin.ac.uk

Active: ☒

Validator: ☐

Administrator: ☐

Save

Figure 52: Modify user – details

This interface can be used to modify the users details, and the roles of the user; *Active* is the default role, which denotes a workflow developer.

Validator is a validator actor.

Administrator is the highest role.

The *Password* tab can be used to change the users password.

The *Groups* tab can be used to list groups of which the selected user is a member.

The *Owned Groups* tab can be used to list the groups the user owns.

10.4 DELETE USERS

The *Actions* button from Figure 52 can be used to delete the user.

11. GROUP MANAGEMENT

Only administrators can list all groups, and perform modifications on such.

11.1 BROWSE GROUPS

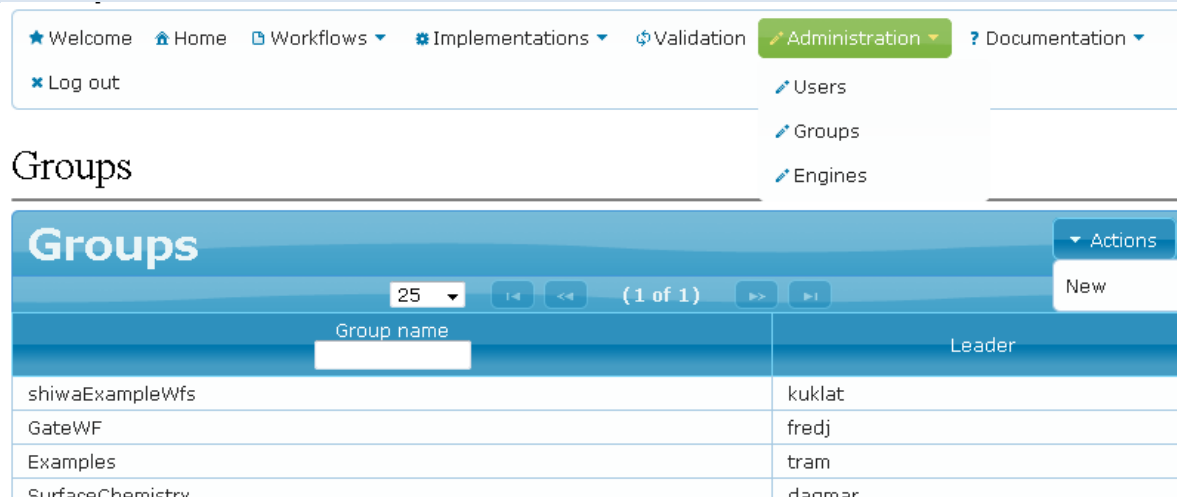


Figure 53: Browse groups

This interface can be used to browse the groups.

11.2 CREATE GROUPS

The *Actions* button from Figure 53 can be used to create new groups.

11.3 MODIFY GROUPS

Groups selected from Figure 53 can be modified.

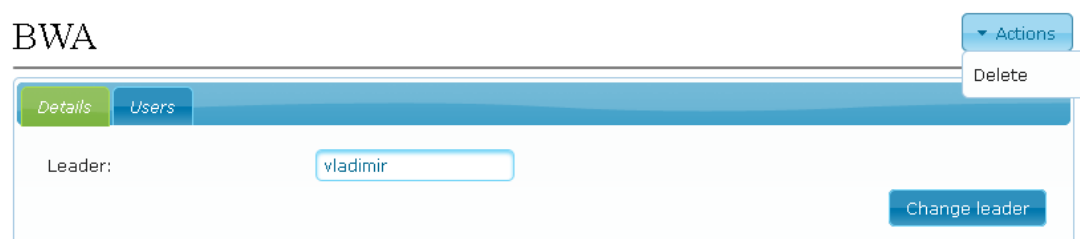


Figure 54: Modify groups

The group leader can be modified.

The *Users* tab can be used to list and modify the group members.

11.4 DELETE GROUPS

The *Actions* button in Figure 54 can be used to delete groups.

12. ENGINE MANAGEMENT

Only administrators may create and manage workflow engines.

Should workflow developers require new workflow engines, they should contact administrators.

12.1 BROWSE ENGINES

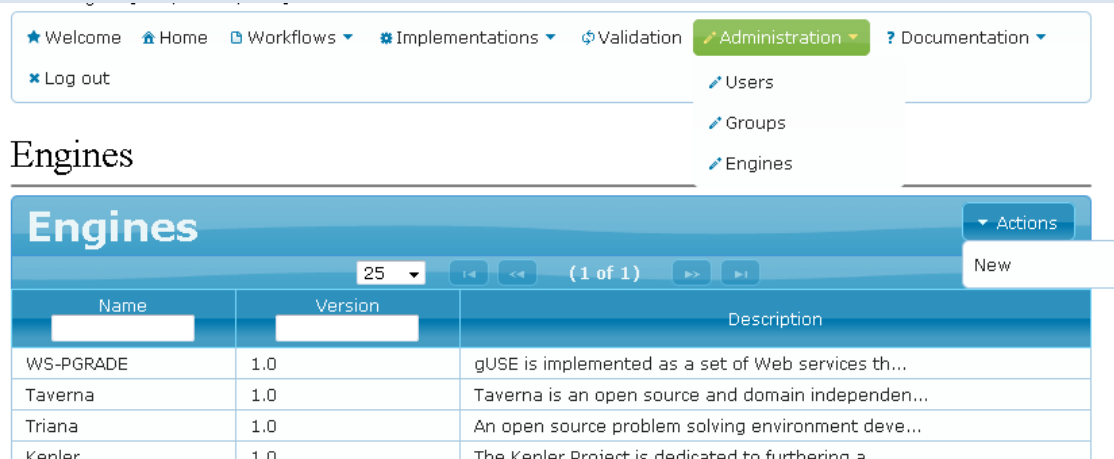


Figure 55: Browse engines

12.2 CREATE ENGINES

Engines may be created, using the Actions button in Figure 55.

12.3 MODIFY ENGINES

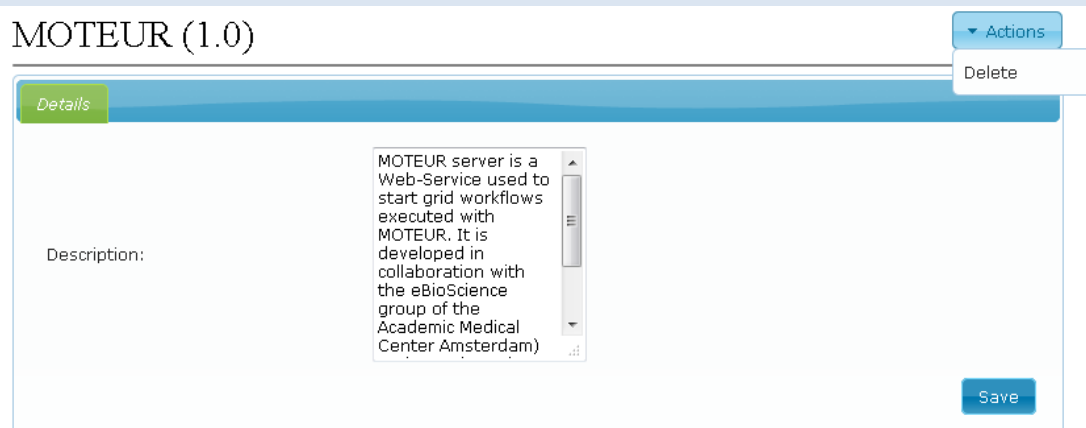


Figure 56: Modify engines

The details of selected engines from Figure 55 may be modified.

12.4 DELETE ENGINES

Engines may be deleted, using the Actions button from Figure 56

13. SERVLET INTERFACE

The SHIWA Repository Servlet Frontend allows upload and download of entities such as workflows, implementations, and data sets in a zip based bundle file format, called SHIWA Bundle, which is a mechanism for physically aggregating resources in a zip file for publishing and archiving. The SHIWA Repository Servlet Frontend was mainly designed to enable communication and workflow exchange between the SHIWA Repository and the SHIWA Desktop. Further information on the SHIWA Bundle and the SHIWA Desktop can be found on the following URL:

<https://www.shiwa-workflow.eu/wiki/-/wiki/Main/SHIWA+Desktop>.

The frontend consists of four servlets: ListContents, ValidateSignature, DownloadBundle, and UploadBundle.

ListContents allows to gather basic information about the Workflows, their Implementations and Configurations.

DownloadBundle allows to download a Workflow, an Implementation or a Configuration either separately or in a single bundle file.

UploadBundle allows to upload a Workflow, an Implementation or a Configuration either separately or in a single bundle file.

SignatureValidation allows to check if the signature of an Implementation meets the signature of the Workflow which it will be added to. This is to be checked before an Implementation Bundle is uploaded to the Repository.

The followings describe in detail the functionality of these servlets and provide usage examples.

13.1 LISTING KEY INFORMATION ABOUT REPOSITORY ITEMS

function:

list summaries of workflows user can read/download

servlet:

org.shiwa.repository.toolkit.servlet.ListContents

usage example:

http://uname:passwd@hostname/shiwa-repo/workflows

returns:

with a list of Workflow Summary Objects containing workflow_id, workflow_name, workflow_description, keywords if successful, error message otherwise

function:

list summaries of workflows user can modify

servlet:

org.shiwa.repository.toolkit.servlet.ListContents

usage example:

http://uname:passwd@hostname/shiwa-repo/workflows/modify

returns:

with a list of Workflow Summary Objects containing workflow_id, workflow_name, workflow_description, keywords if successful, error message otherwise

function:

list summaries of implementations of a particular workflow

SHIWA Workflow Repository – Administrator Manual

servlet:

org.shiwa.repository.toolkit.servlet.ListContents

usage example:

http://uname:passwd@hostname/shiwa-repo/workflows/1002/imps

returns:

with a list of Implementation Summary Objects containing implementation_id, workflow_id, implementation_version, engine_name, engine_version, title, description, keywords, \verbDCIs+ if successful, error message otherwise

function:

list summaries of configurations of a particular workflow

servlet:

org.shiwa.repository.toolkit.servlet.ListContents

usage example:

http://uname:passwd@hostname/shiwa-repo/workflows/1002/confs

returns:

with a list of Configuration Summary Objects containing configuration_id, description if successful, error message otherwise

13.2 SIGNATURE VALIDATION

function:

check whether the given signature is valid

servlet:

org.shiwa.repository.toolkit.servlet.ValidateSignature, input file: signature file

usage example:

http://uname:passwd@hostname/shiwa-repo/validatesignature

returns:

with an accepted message if signature is valid, invalidation details otherwise Upload bundle

function:

upload a workflow/implementation/configuration - depending on the contents of the bundle file (with validation check)

servlet:

org.shiwa.repository.toolkit.servlet.UploadBundle

input file:

bundle file

usage example:

http://uname:passwd@hostname/shiwa-repo/uploadbundle

returns:

with an accepted message if signature is valid and upload is successful, error message otherwise

function:

upload a workflow/implementation/configuration - depending on the contents of the bundle file (ignore validation check)

servlet:

org.shiwa.repository.toolkit.servlet.UploadBundle

input file:

bundle file

usage example:

http://uname:passwd@hostname/shiwa-repo/uploadbundle/force

returns:

with an accepted message if upload is successful, error message otherwise

13.3 DOWNLOAD BUNDLE

function:

download a workflow

servlet:

org.shiwa.repository.toolkit.servlet.DownloadBundle

usage example:

http://uname:passwd@hostname/shiwa-repo/downloadbundle/1002

returns:

SHIWA Workflow Repository – Administrator Manual

with a bundle file containing the workflow and related metadata if successful, error message otherwise

function:

download an implementation

servlet:

org.shiwa.repository.toolkit.servlet.DownloadBundle

usage example:

http://uname:passwd@hostname/shiwa-repo/downloadbundle/1002/2051

returns:

with a bundle file containing an implementation and related metadata if successful, error message otherwise

function:

download an implementation with parent workflow

servlet:

org.shiwa.repository.toolkit.servlet.DownloadBundle

usage example:

http://uname:passwd@hostname/shiwa-repo/downloadbundle/1002/2051?wf=true

returns:

with a bundle file containing an implementation, plus its parent workflow and related metadata if successful, error message otherwise

function:

download a workflow with implementations

servlet:

org.shiwa.repository.toolkit.servlet.DownloadBundle

usage example:

http://uname:passwd@hostname/shiwa-repo/downloadbundle/1002?imps=2051,2052,2053

returns:

with a bundle file containing the workflow, the requested set of implementations and related metadata if successful, error message otherwise

function:

download a workflow with configurations

servlet:

org.shiwa.repository.toolkit.servlet.DownloadBundle

usage example:

http://uname:passwd@hostname/shiwa-repo/downloadbundle/1002?confs=1,2,3

returns:

with a bundle file containing the workflow, the requested set of configurations and related metadata if successful, error message otherwise

function:

download a workflow with implementations and configurations

servlet:

org.shiwa.repository.toolkit.servlet.DownloadBundle

usage example:

http://uname:passwd@hostname/shiwa-repo/downloadbundle/1002?imps=2051,2052,2053&confs=1,2,3

returns:

with a bundle file containing the workflow, the requested set of implementations, configurations and related metadata if successful, error message otherwise

14. REPOSITORY METADATA GRAPH (FIGURE 57)

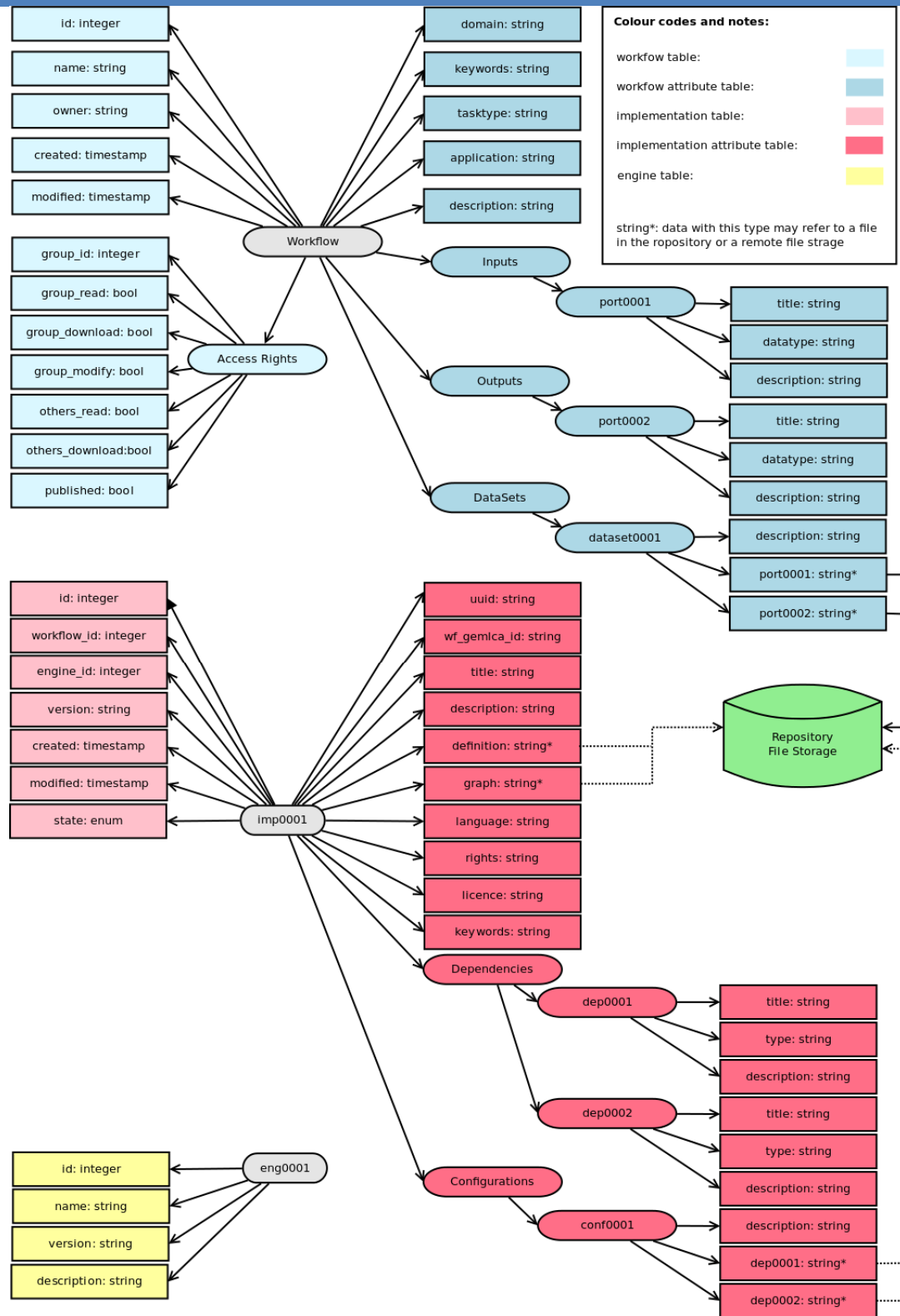


Figure 57: SHIWA Repository metadata graph

15. LIMITATIONS

- It is not recommended to open the repository in multiple browser tabs.
- Concurrent editing is currently not supported