Eclipse Winery

Contributors to the Eclipse Foundation

Jul 06, 2022

CONTENTS:

1	User Guide	3
2	Developer Guide	31
3	Notes on TOSCA	33
4	Architectural Decision Log	35
5	Getting support for Eclipse Winery	37
6	License	39

Eclipse Winery is a web-based environment to graphically model TOSCA topologies and plans managing these topologies. The environment includes a type and template management component to offer creation and modification of all elements defined in the TOSCA specification. All information is stored in a repository, which allows importing and exporting using the TOSCA packaging format.

Are you tired of maintaining your TOSCA files manually by just using a text editor?

Use Eclipse Winery as an usability layer on top to maintain your TOSCA files (XML or YAML) in a graphical and intuitive user interface. Eclipse Winery provides a graphical web-editor with which you can create and maintain all TOSCA entities. Thereby, Eclipse Winery stores all TOSCA entities in a defined folder structure that fosters the reusability of TOSCA types. Eclipse Winery validates and stores all TOSCA entities in the syntax defined in the standard.

Further, the graph-based representation of TOSCA topologies in Eclipse Winery provides a quick overview of the entire system and offers a communication basis for the cooperation with other parties. It therefore offers a quicker introduction to modeling with TOSCA and provides newcomers with necessary guidelines.

This is the main documentation of Eclipse Winery.

Organizational information is provided at the eclipse.org page.

Demo	Video
Getting Started	User Guide
Licence	EPL-2.0 OR Apache-2.0
Maintainer(s)	Eclipse Winery Contributors

CHAPTER

ONE

USER GUIDE

Eclipse Winery is a web-based environment to graphically model *OASIS TOSCA* topologies and plans managing these topologies. It is an Eclipse project and thus support is available through its project page. Winery is also part of the OpenTOSCA ecosystem where more information is available at opentosca.org. For more information on TOSCA see our *TOSCA information page*.

1.1 Getting Started

1.1.1 Launching with Docker

Note: It is recommended that your host or virtual machine has at least 2GB of memory.

Open a command prompt and execute the following command:

```
docker run -it -p 8080:8080 \
   -e PUBLIC_HOSTNAME=localhost \
   -e WINERY_FEATURE_RADON=true \
   -e WINERY_REPOSITORY_PROVIDER=yaml \
   -e WINERY_REPOSITORY_URL=https://github.com/radon-h2020/radon-particles \
   opentosca/radon-gmt
```

Launch a browser: http://localhost:8080.

Note: To start Eclipse Winery based on an TOSCA XML repository layout, use the following command:

```
docker run -it -p 8080:8080 \
    -e PUBLIC_HOSTNAME=localhost \
    -e WINERY_REPOSITORY_URL=https://github.com/OpenTOSCA/tosca-definitions-public \
    opentosca/winery
```

Note: Make sure you regularly pull the latest images:

```
docker pull opentosca/radon-gmt:latest
# or
docker pull opentosca/winery:latest
```

Use a custom TOSCA model repository

Problem: You want to use an existing TOSCA model repository that you have cloned, e.g., to add new or adapt existing TOSCA types and blueprints in this Git repository.

Please follow the next instructions to mount an existing TOSCA model repository into the Eclipse Winery container. This is useful if you want to save your modeling changes onto your Docker host machine.

Clone or create git repository on your local filesystem, e.g., by cloning https://github.com/radon-h2020/radon-particles.

Open a command prompt and execute the following command:

Warning: Replace con_your_host> with the respective directory path on your host system.

```
docker run -it -p 8080:8080 \
  -e PUBLIC_HOSTNAME=localhost \
  -e WINERY_FEATURE_RADON=true \
  -e WINERY_REPOSITORY_PROVIDER=yaml \
  -v <path_on_your_host>:/var/repository \
  -u `id -u` \
  opentosca/radon-gmt
```

Launch a browser: http://localhost:8080.

Any change (create service template, modify or create node types) will be reflected on your host machine. You are now able to commit your changes and push them to your own Git remote (e.g., using git push from a command-prompt).

Note: To start Eclipse Winery based on an TOSCA XML repository layout, use the following command:

```
docker run -it -p 8080:8080 \
  -e PUBLIC_HOSTNAME=localhost \
  -v <path_on_your_host>:/var/repository \
   opentosca/winery
```

Model and version your applications and company-specific types only

Problem: You want to model applications based on actively maintained TOSCA type repositories but you want to version/save only your own application blueprints and company-specific types inside your (private) Git repository (GitHub, GitLab).

You are able to start Eclipse Winery in a so-called "multi-repository" setup where you can add several TOSCA type repositories that you can use for your application models. However, with this setup, Eclipse Winery creates a specific "workspace" that only contains your company-specific types and application blueprints (separated by namespace). You can then mount the created "workspace" to save your modeling results to your own Git remote.

Open a command prompt and execute the following command:

Warning: Replace <path_on_your_host> with a respective dirctory path on your host system.

```
docker run -it -p 8080:8080 \
    -e PUBLIC_HOSTNAME=localhost \
    -e WINERY_FEATURE_RADON=true \
    -e WINERY_REPOSITORY_PROVIDER=yaml \
    -e WINERY_DEPENDENT_REPOSITORIES="[ { \"name\" : \"RADON Particles\", \"url\" : \
    -"https://github.com/radon-h2020/radon-particles.git\", \"branch\" : \"master\" } ]"_
    +\
    -v <path_on_your_host>:/var/repository \
    -u `id -u` \
    opentosca/radon-gmt
```

Your created TOSCA service templates or company-specific TOSCA node types will be stored on your host machine. You are now able to commit your changes and push them to your own Git remote (e.g., using git push from a command-prompt).

1.1.2 Launching with Docker Compose

Note: It is recommended that your host or virtual machine has at least 2GB of memory.

Install Docker and Docker Compose.

Clone the repository:

```
git clone https://github.com/eclipse/winery
cd winery/deploy/compose
```

[Optional] Adapt the Docker Compose configuration to your needs, e.g., to mount a local TOSCA model repository.

Start Winery:

docker-compose up

Launch a browser: http://localhost:8080.

1.2 Modeling with Winery

Launch a browser and navigate to http://localhost:8080.

1.2.1 Modeling an Application

Eclipse Winery starts in the *Service Template* view. In this view, users can create new TOSCA service template or maintain existing ones.

To create a new TOSCA service template click on *Add new*. In the "Add new" pop up you can specify your template's name, enable/disable versioning, and specify the namespace to be used. For example, you may choose a namespace like com.example.blueprints to logically group your TOSCA service templates.

Warning: Do not use spaces in your service template name. Use _ or – to separate names.

In the *Service Template Detail* view you can add some readme text and assign a respective license. Further, to compose your application open the *Topology Modeler* by Topology Template > Open Editor.

Model Node Templates

In the editor, you can drag and drop existing TOSCA node types to the canvas to define a new TOSCA node template. You can select a modeled node to modify its display name and additional data using the right pane.

You can change properties or add artifacts by enabling the Properties or Artifacts view in the header bar.

Define Relations Between Node Templates

Relationships in TOSCA (according to TOSCA YAML 1.3) are modeled using matching *Requirements* and *Capabilities* (please refer to the standard to get more detailed information or checkout the *Notes on TOSCA* page).

In the *Topology Modeler*, you can enable the *Requirements & Capabilities* view in the header bar. Then, open the *Requirements* of the source node and the *Capabilities* of the target node. Finally, drag a respective relationship type (e.g., HostedOn) from the requirement (e.g., host) to a matching capability (e.g., host).

1.2.2 Export CSAR

The TOSCA exchange format is a Cloud Service Archive (CSAR). A CSAR is essentially a ZIP file following a certain directory layout and contains all necessary files and template to execute the deployment of the modeled application.

Open the *Service Template* view. Search for your service template and open it. In the *Service Template Detail* view you can click on *Export* either to *Download* the CSAR or to save it to the filesystem (<repository>/csars on your host system).

1.3 Node Type Development

Start Eclipse Winery as described in "Use a custom TOSCA model repository" of the Getting Started page. This way, newly created Node Types will be reflected in the filesystem which is mounted into Winery's Docker container.

Before you start, create a new branch:

```
git checkout -b <name>
```

You can push this branch to a Git origin to share your work with others or you could propose a pull-request to the original Git repository.

1.4 Modeling based on TOSCA XML (deprecated)

This guide shows an overview of how to model TOSCA node types and service templates using Winery. Before starting this guide, please take a look at *Miscellaneous Notes*.

The following shows how to model new node types and how to use them at the modeling of a new service template. In this example, the runtime **Python3** shall be installed on an **Ubuntu 14.04** virtual machine running on an **Openstack** infrastructure. For this, we require three node types. In this example, we model two node types, Python3 and Ubuntu 14.04, and assume that the OpenStack node type was previously modeled.

1.4.1 Creating a new Node Type

By selecting the tab *Node Types*, a list of available node types is shown. To create a new node type, press the button *Add new*.

Winery				about
Service Templates	Node Types Relationship Types	Other Elements A	dministration	
				Add new
				Import CSAR
http	p://opentosca.org/nodetypes	52		Import YAML
				Show all Items
http	p://opentosca.org/nodetypes/versioned	6		
	Items per Page: 10 v « Previous	1 Next »		

This will open a dialog in which the *Name*, *Component version*, and *Namespace* of the new node type can be configured.

Winery		about
Service Templates No	Add a new Node Type ×	
	Name	Add new
	Python	Import CSAR
http://open	Versioning:	Import YAML
		Show all Items
http://open	The component version specifies the components' external version defined by the creator of the software (e.g., Apache Tomcat 8.5.1 has a component version 8.5.1). Winery adds management to the software which is versioned independently of the softwares' version. The version inside Winery is called management version and is mandatory. It consists of a winery version and a work in progress (wip) version. Upon the creation of a new component, both management versions are set automatically with their initial values of 1. The generated name is displayed in the 'Final name' field (e.g., the final id for Apache Tomcat 8.5.1 is Tomcat_8.5.1-v1-vip1).	
	When developing a TOSCA definition, a the wip version is appended until the TOSCA	
	definition is stable. To test a TOSCA definition, the wip version can be committed. After a version was committed, a new version must be added to apply further changes. Thus, a new wip version must be added (e.g., Tomcat_8.5.1-w1-wip1 followed by Tomcat_8.5.1-w1-wip2 released as Tomcat_8.5.1-w1 must be followed by Tomcat_8.5.1-w2-wip1 to enable changes). Thereby, different component versions do not affect each other (e.g., Tomcat_8.5.1-w2-wip1 can be created while Tomcat_9.0.1-w3 exists).	
	Final name	
	Python_3-w1-wip1	
	Namespace	
	http://opentosca.org/nodetypes/versioned	
	Template:	
	Cancel Add	

Once the node type is created, it can be further configured through different tabs of its detailed view.

Winery							
rvice Templates	Node Ty	pes	Relationship Ty	ypes Other Elemen	ts Administr	ation	
	NC Version: opentosca.or	and the second second second	es		De	elete Export -	Versions +
README L	ICENSE	Visua	I Appearance	Instance States	Interfaces	Implementations	Tags
Requirement Definit	tions	Capability	/ Definitions	Properties Definition	Inheritance	Documentation	XML
66 This Node Properties	6						
Properties None. Haftungsa Dies ist ein Forschungspi insbesondere bei variiertu sämtlicher Lizenzen gepri	S AUSSCI rototyp und en en oder neuer üft werden. Die	nluss nthält insbeso n Anwendung e Haftung für	ondere Beiträge von jsfällen, nicht richtig - entgangenen Gewi	n Studenten. Diese Software en g. Insbesondere beim Produktive inn, Produktionsausfall, Betriebs und Folgeschäden ist, außer i	einsatz muss 1. die Funl sunterbrechung, entgan	ktionsfähigkeit geprüft und 2 ngene Nutzungen, Verlust vo	. die Einhaltung n Daten und

For example, to add properties to the node type, select the tab *Properties Definition*.

Winery						
ervice Templates	Node Types	Relationship Typ	oes Other Element	s Admini	stration	
	Version: 3-w1-w opentosca.org/nodel				Delete Export •	Versions +
README L	ICENSE Vis	ual Appearance	Instance States	Interfaces	Implementations	Tags
Requirement Defini	tions Capab	ility Definitions	Properties Definition	Inheritance	Documentation	XML
◎ (none)	irs					Save
	A CONTRACTOR OF THE OWNER OWNER OF THE OWNER OWNE OWNER OWNE					

In this example, the Python3 node type does not require any properties.

1.4.2 Modeling the Node Type Interface

To specify what the *Python3* node type should do, we define an *interface* and the *operations* provided by this interface. An interface containing lifecycle operations (install, configure, start, stop, uninstall) can be automatically generated, however, any arbitrary interface can be created.

To generate a lifecycle interface, press *Generate Lifecycle Interface* and *Save*. For the node type Python3, we only use the operation *install*.

Winery						
ervice Templates	Node Types	Relationship T	ypes Other Elemer	ts Administ	tration	
	Version: 3-w1-w pentosca.org/node			C	elete Export -	Versions -
README L	ICENSE Vi	sual Appearance	Instance States	Interfaces	Implementations	Tags
Requirement Definit	tions Capat	ility Definitions	Properties Definition	Inheritance	Documentation	XML
Interfaces http://www.example.com/	Add Remove interfaces/lifecycle	Operations Install	Add Remove			Save
Generate Implementation	on Artifact Generat	e Lifecycle Interface				
	No. of Concession, Name					

1.4.3 Modeling an Artifact Template for a Node Type Operation

Once the operations of a node type are defined, artifacts (e.g., shell scripts, .war files) implementing these operations need to be modeled. In this example, we have a *shell script* to install Python3 on Ubuntu, which we model as an artifact template.

To create an artifact template, select the tab *Other Elements*, under the category *Artifacts* select the option *Artifact Templates*, and press the button *Add new*.

Winery					
Service Templates	Node Types	Relationsh	ip Types	Other Elements	Administration
The following items list T	OSCA elements co	ntained in TOSC	A's Definitio	ns element, which ar	e not listed as separate tabs
Artifacts					
Artifact Typ	es	Artifact Te	mplates		
Requirements and	Capabilities				
Requirement	Types	Capabilit	y Types		
Implementations					
Node Type Implem	nentations R	elationship Type	Implementatio	ons	
Policies					
Policy Type	es	Policy Te	mplates		
Imports					
XML Schema De	efinitions	WSE)Ls		
Compliance Rules					
Compliance F	Rules				
Winery					ab
	Node Types Rela	ationship Types	Other Element	s: Artifact Templates	Administration
					Add new
					Import CSAR
Admin http://op	entosca.org/artifactten	nplates		94	Import YAML
					Show all Items
Admin//open	ntosca.org/artifacttemp	lates/versioned		10	
Item	ns per Page: 10 🗸	« Previous ⁴	Next »		

This will open a dialog in which the *Name*, *Versioning*, *Type*, and *Namespace* of the artifact template can be configured. Assuming that some artifact types were previously modeled, choose the type *ScriptArtifact*.

Winery			about
Service Templates No	Add a new Artifact Template	×	tration
	Name		Add new
	Python_3_Install_IA		Import CSAR
Admin http://open	Versioning:	~	Import YAML
	Туре		Show all Items
	ScriptArtifact	•	
Admin//opento	Namespace		
	http://opentosca.org/artifacttemplates		
Items	Template:	~	
	Cance	l Add	

Once the artifact template is created, it can be further configured through different tabs of its detailed view.

ervice Templates	Node Types	Relationship Types	Other Elements: Artifact Templates	Administration	
Admin	on_3_Install_IA® Ver /opentosca.org/artifac			Delete Export -	Versions -
README	LICENSE File	es Source F	Properties Property Constraints	Documentation	XML
Python	3 Instal		e 2 0		de la calegra
-		License Apache	e 2.0		Ø
Haftungs	ausschlus	S			8
Haftungs Dies ist ein Forschung insbesondere bei variid sämtlicher Lizenzen ge	ausschlus sprototyp und enthält inst erten oder neuen Anwend sprüft werden. Die Haftung	S besondere Beiträge von Student lungsfällen, nicht richtig. Insbeso g für entgangenen Gewinn, Prov	e 2.0 ten. Diese Software enthält möglicherweise Fehle ondere beim Produktiveinsatz muss 1. die Funkti duktionsausfall, Betriebsunterbrechung, entgang olgeschäden ist, außer in Fällen von grober Fahr	onsfähigkeit geprüft und 2. d ene Nutzungen, Verlust von I	ie Einhaltung Daten und
Haftungs Dies ist ein Forschung insbesondere bei varili sämtlicher Lizenzen ge Informationen, Finanzi ausgeschlossen.	ausschlus sprototyp und enthält inst erten oder neuen Anwend sprüft werden. Die Haftung	S besondere Beiträge von Student lungsfällen, nicht richtig. Insbes g für entgangenen Gewinn, Pro vie sonstige Vermögens- und Fo	ten. Diese Software enthält möglicherweise Fehle ondere beim Produktiveinsatz muss 1. die Funkti duktionsausfall, Betriebsunterbrechung, entgang	onsfähigkeit geprüft und 2. d ene Nutzungen, Verlust von I	ie Einhaltung Daten und

Finally, to load th install script to the artifact template, select the tab *Files*, and drop the file into the drop zone.

Winery						
Service Templates	Node Types	Relationsh	ip Types Othe	r Elements: Artifact Templates	Administration	
	ython_3_Install_IA® tp://opentosca.org/ar				Delete Export - Type: ScriptArtifact	Versions -
README	LICENSE	Files Sour	ce Properties	Property Constraints	Documentation	XML
Durchsuchen You may dro	Keine Dateien ausgewi	ählt.				ZIP
The files are imme Contained Files	diately uploaded withou	t any confirmation.				
install.sh	0.07 KB Delete					

1.4.4 Modeling the Node Type Implementation

To create a node type implementation, select the tab *Other Elements*, under the category *Implementations* select the option *Node Type Implementations*, and press the button *Add new*. This will open a dialog in which the *Name*, the corresponding node *type*, and *Namespace* of the node type implementation can be configured. By type, select the node type we created before.

Winery			about
Service Templates No	Add a new Node Type Implementation	×	Administration
	Name		Add new
	Python_3_Impl		Import CSAR
Admin Apache-2.4 http://opentos	Versioning:	~	Import YAML
	Туре		Group by Namespace
Apache-Sp	Python_3-w1-wip1	•	
Admin http://opentos	Namespace		
Apache-Sp	http://opentosca.org/nodetypeimplementations		
Admin http://opentos	Template:	~	
Admin DockerCor http://opentos	Cancel	Add	

Service Templates	Node Types	Relationship Types	Other Elements: Node	Type Implementatio	Administratio	n
Admin	non_3_Impl Versio //opentosca.org/nod	n: w1-wip1 letypeimplementations@			elete Export - Ve ementation for Python_	ersions - 3-w1-wip
README	LICENSE	mplementation Artifacts	Deployment Artifacts	Inheritance	Documentation	XML
Haftungs	ausschlu	p ss	License Apache 2.0			
Haftungs Dies ist ein Forschung insbesondere bei variid sämtlicher Lizenzen g	sprototyp und enthält ir erten oder neuen Anwen eprüft werden. Die Haftu		ten. Diese Software enthält mögli ondere beim Produktiveinsatz mus duktionsausfall, Betriebsunterbrec	ss 1. die Funktionsfähig hung, entgangene Nutz	keit geprüft und 2. die Einha ungen, Verlust von Daten ur	nd
Haftungs Dies ist ein Forschung insbesondere bei variti sämtlicher Lizenzen g Informationen, Finanzi ausgeschlossen.	sprototyp und enthält ir erten oder neuen Anwen eprüft werden. Die Haftu	SS nsbesondere Beiträge von Student Idungsfällen, nicht richtig. Insbeso ung für entgangenen Gewinn, Proc owie sonstige Vermögens- und Fo	ten. Diese Software enthält mögli ondere beim Produktiveinsatz mus duktionsausfall, Betriebsunterbrec	ss 1. die Funktionsfähig hung, entgangene Nutz	keit geprüft und 2. die Einha ungen, Verlust von Daten ur	nd

To link the created artifact template to this node type implementation, select the tab *Implementation Artifacts* and press the button *Add*. In the shown dialog, choose the option *Link Artifact Template*, then select the artifact template that was previously created.

Winery			about
Service Templates No	Add Implementation Artifact	×	Administration
Admin Python_3_in http://opento README LICENS	Name Python_3_Impl_IA Interface Name http://opentosca.org/interfaces/lifecycle Operations	·	Export Versions on for Python_3-w1-wip1 umentation XML
Available Implementation An	Install Artifact Template Creation	~	
Search: Filter all columns	Create Artifact Template Check if you want to upload new files, you do not want to reuse existing files and you do not point to an image library. Eink Artifact Template	1	Add Remove
Name Artifat	Check if you want to reuse existing files. O Do not create an artifact template Check if you want to point to an image library. Artifact Template		
	Python_3_Install_IA_w1-wip1		×
	Cancel Ad	d	

vice Templates	Node Types	Relationship Types	Other Elements: N	ode Type Implementati	ions Administra	ation
imin	/thon_3_Impl Ver tp://opentosca.org/r	sion: w1-wip1 odetypeimplementations&			Delete Export -	Versions - n_3-w1-wij
README	LICENSE	Implementation Artifacts	Deployment Artifacts	Inheritance	Documentation	XM
Available Implem	entation Artifacts					
wailable Implem Gearch: Filter all colum						
Search:					A	dd Remove
Search:		Artifact Template		Artifact Type	Ad Specific Content	dd Remove

1.4.5 Modeling the Ubuntu Node Type

The modeling of the Ubuntu node type is similar to the modeling of the Python3 node type.

ervice Templates			<u></u>	Contraction of the second		
	Node Types	Relationship Ty	pes Other Elemer	nts Administ	ration	
- Ubur	tu Version: 14.04	1.w1.wip1				Marrian
	/opentosca.org/nod				elete Export -	Versions -
README	LICENSE	Visual Appearance	Instance States	Interfaces	Implementations	Tags
Requirement Defin	nitions Cap	ability Definitions	Properties Definition	Inheritance	Documentation	XML
Ubuntu	14 04	icense Apache 2.0				
Obuntu		icense Apacile 2.0				
66 This nod	e type correspon	ds to an Ubuntu-14	.04 virtual machine.			
-						
 VMIP (optiona VMInstanceID VMType VMUserName VMUserPasswo 	(optional) rd					
 VMPrivateKey VMPublicKey VMKeyPairNam 	e					
• VMPublicKey		SS				
• VMPublicKey • VMKeyPairNam Haftungs Dies ist ein Forschung: insbesondere bei variie samtlicher Lizenzen ge Informationen, Finanzie	ausschlu sprototyp und enthält in rten oder neuen Anwer sprüft werden. Die Haftu	nsbesondere Beiträge von ndungsfällen, nicht richtig. ung für entgangenen Gewi	Studenten. Diese Software en Insbesondere beim Produktive nn, Produktionsausfall, Betriet und Folgeschäden ist, außer	insatz muss 1. die Fur sunterbrechung, entga	ktionsfähigkeit geprüft und 2 ngene Nutzungen, Verlust vo	. die Einhaltung n Daten und
• VMPublicKey • VMKeyPairNam Haftungs Dies ist ein Forschung: insbesondere bei variie sämtlicher Lizenzen ge Informationen, Finanzie ausgeschlossen.	ausschlu sprototyp und enthält in tren oder neuen Anwer sprüft werden. Die Haftu rrungsaufwendungen so	nsbesondere Beiträge von ndungsfällen, nicht richtig. ung für entgangenen Gewi owie sonstige Vermögens-	Insbesondere beim Produktive nn, Produktionsausfall, Betrieb	insatz muss 1. die Fur sunterbrechung, entga	ktionsfähigkeit geprüft und 2 ngene Nutzungen, Verlust vo	. die Einhaltung n Daten und
VMPublicKey VMKeyPairNam Maftungs Dies ist ein Forschung: insbesondere bei variie sämtlicher Lizenzen ge Informationen, Finanzie ausgeschlossen. Disclaimee	ausschlu sprototyp und enthält in rten oder neuen Anwer sprüft werden. Die Haftu rrungsaufwendungen so er of War	nsbesondere Beiträge von ndungsfällen, nicht richtig. ung für entgangenen Gewi owie sonstige Vermögens- ranty	Insbesondere beim Produktive nn, Produktionsausfall, Betrieb	insatz muss 1. die Fur sunterbrechung, entga in Fällen von grober Fa	ktionsfähigkeit geprüft und 2 ngene Nutzungen, Verlust vo hrlässigkeit, Vorsatz und Pe	t. die Einhaltung n Daten und rsonenschäden

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

rvice Templates	Node Types	Relationship	Types	Other Eleme	nts	Administ	tration	
	u Version: 14. opentosca.org/n					D	elete Export -	Versions -
README L	ICENSE	Visual Appearance	Insta	nce States	Int	erfaces	Implementations	Tags
Requirement Definit	tions Ca	apability Definitions	Proper	ties Definition		Inheritance	Documentation	XML
○(none) ○XML element ○XML type ●Custom key/value pa	airs							Save
Properties Wrap								
Properties vvra	oper							
	oper							
	oper							Add Remove
Search:	sper					Туре		Add Remove
Search: Filter all columns	oper					Type xsd:string		Add Remove
Search: Filter all columns Name	sper							Add Remove
Search: Filter all columns Name VMIP	sper					xsd:string		Add Remove
Search: Filter all columns Name VMIP VMInstanceID	sper					xsd:string xsd:string		Add Remove
Search: Filter all columns Name VMIP VMInstanceID VMType	pper					xsd:string xsd:string xsd:string		Add Remove
Search: Filter all columns Name VMIP VMInstanceID VMType VMUserName	pper					xsd:string xsd:string xsd:string xsd:string		Add Remove
Search: Filter all columns Name VMIP VMInstanceID VMType VMUserName VMUserPassword	sper					xsd:string xsd:string xsd:string xsd:string xsd:string		Add Remove

However, the artifact templates for the Ubuntu 14.04 are *.war files* instead of *shell scripts*. In this case, after defining the *interfaces* and *operations* of the Ubuntu node type, we can automatically generate a stub java maven project to build a *.war file* for a defined interface. For this, press *Generate Implementation Artifact*. The node type implementation will be automatically generated as well.

Templates Node Types Relationship Types Other Elem Ubuntu© Version: 14.04-w1-wip1 http://opentosca.org/nodetypes© Instance States DME LICENSE Visual Appearance Instance States uirement Definitions Capability Definitions Properties Definition aces Add Remove Operations Add Remove opentosca.org/interface installPackage installPackage rate Implementation Artifact Generate Lifecycle Interface Parameters xsd:String wateKey xsd:String Parameters String			
http://opentosca.org/nodetypes DME LICENSE Visual Appearance Instance States Properties Definition aces Add Add Remove Operations Add ingSystemInterface installPackage transferFile umScript unsteriot Generate Lifecycle Interface Parameters serName xsd:String ivateKey xsd:String	ents Adminis	stration	
airement Definitions Capability Definitions Properties Definition acces Add Remove installPackage transferFile runsfrigt waitForAvailability rate Implementation Artifact Generate Lifecycle Interface Parameters serName serName serName setString se	ſ	Delete Export -	Versions -
aces Add Remove Operations Add Remove installPackage transferFile runScript waitForAvailability aratel Implementation Artifact Generate Lifecycle Interface Parameters serVame xsd:String xsd:String invateKey xsd:String xsd:String	Interfaces	Implementations	Tags
Perturbis installPackage installtitue installPackage installPackage installPackage	Inheritance	Documentation	XML
xsd:String xsd:String ivateKey xsd:String			Add Remove
serName xsd:String ivateKey xsd:String		Required	
ivateKey xsd:String		YES	
		YES	
Parameters		YES	
			Add Remove
Туре		Required	
Result xsd:String		YES	

Winery		_	ab
ervice Templates No	Generate Implementation Artifact ×		
Ubuntu@ Ve	Java Package		Versions -
http://opento	org.opentosca.nodetypeimplementations		
README LICENS	Node Type Implementation Name	6	Tags
Requirement Definitions	Ubuntu-14.04-w1-wip1-Implementation	tion	XML
Interfaces Add	Will be created.		Save
OperatingSystemInterface http://opentosca.org/interfaces/	Final name		
	Ubuntu-14.04-w1-wip1-Implementationw1-wip1		
	Namespace		
Generate Implementation Artif	http://opentosca.org/nodetypeimplementations		
Input Parameters	There is no check for the name of the implementation artifact. The artifact template name will be reused as implementation artifact name without any further check.		Add Remove
Name	Artifact Template Name		
VMIP	Ubuntu-14.04-w1-wip1-OperatingSystemInterface-IA		
VMUserName VMPrivateKey	Will be created.		
Output Parameters	Final name		Add Remove
	Ubuntu-14.04-w1-wip1-OperatingSystemInterface-IAw1-wip1		Add Kentove
Name WaitResult	Namespace		
	http://opentosca.org/artifacttemplates		
			_
	Cancel Generate		

Vinery								
rvice Templates	Node Type:	s Relationship Type	es Oth	er Elements	Administ	ration		
	Version: 14 pentosca.org/r				De	elete	ort -	Versions -
	OFNOF		lester o		h fa a	less less station	~	Teres
README LI Requirement Definiti	CENSE	Visual Appearance apability Definitions	Instance S Properties D		Inheritance	Implementatio		Tags
This page shows Node Ty repository.	pe Implementati	ons available for this type. Go to	o Node Type Imp	plementations to	get an overview on	all Node Type Impl	ementations	stored in this
Search: Filter all columns								
								Add Remove
Namespace			N	lame				
				Ibuntu-14 04-w1-	vip1-Implementatio	n w1-wip1		
http://opentosca.org/noc	letypeimplement	ations						
http://opentosca.org/noc	letypeimplement	ations						
Vinery	letypeimplement			_	lode Type Imple		Admini	istration
Vinery rvice Templates dmin Ubuntu	Node Types		es Oth n: w1-wip1	_	lode Type Imple De	ementations Hete Expo	rt -	Versions +
Winery rvice Templates dmin Ubuntu http://op	Node Types -14.04-w1-wip pentosca.org/r	s Relationship Type 1-Implementation Versio nodetypeimplementations	es Oth n: w1-wip1	er Elements; M	lode Type Imple De Imple	ementations elete Expo ementation for L	rt - Ubuntu_14	Versions + .04-w1-wip1
Vinery vice Templates dmin Ubuntu http://op	Node Types	s Relationship Type 11-Implementation @ Versio	es Oth n: w1-wip1	_	lode Type Imple De Imple	ementations elete Expo ementation for L	rt -	Versions + .04-w1-wip1
Vinery rvice Templates dmin Ubuntu http://og README LI	Node Types -14.04-w1-wip poentosca.org/r CENSE	s Relationship Type 1-Implementation Versio nodetypeimplementations	es Oth n: w1-wip1	er Elements; M	lode Type Imple De Imple	ementations elete Expo ementation for L	rt - Ubuntu_14	Versions + .04-w1-wip1
Vinery vice Templates dmin Ubuntu http://op README LIU Available Implementati	Node Types -14.04-w1-wip poentosca.org/r CENSE	s Relationship Type 1-Implementation Versio nodetypeimplementations	es Oth n: w1-wip1	er Elements; M	lode Type Imple De Imple	ementations elete Expo ementation for L	rt - Ubuntu_14	Versions + .04-w1-wip1
Vinery vice Templates dmin Ubuntu http://op README LIU Available Implementati	Node Types -14.04-w1-wip poentosca.org/r CENSE	s Relationship Type 1-Implementation Versio nodetypeimplementations	es Oth n: w1-wip1	er Elements; M	lode Type Imple De Imple	ementations elete Expo ementation for L	rt - Ubuntu_14	Versions + .04-w1-wip1
Vinery vice Templates dmin Ubuntu http://op README LII Available Implementati	Node Types -14.04-w1-wip poentosca.org/r CENSE	s Relationship Type 1-Implementation Versio nodetypeimplementations	es Oth n: w1-wip1	er Elements; M	lode Type Imple De Imple	ementations elete Expo ementation for L	rt - Ubuntu_14	Versions + .04-w1-wip1
Vinery rvice Templates dmin Ubuntu http://op README Lli Available Implementati	Node Types -14.04-w1-wip poentosca.org/r CENSE	s Relationship Type 1-Implementation Versio nodetypeimplementations	es Oth n: w1-wip1	er Elements; M	lode Type Imple De Impl s Inherit	ementations elete Expo ementation for L	rt - Ubuntu_14	Versions - .04-w1-wip1 on XML

After editing the generated stub project, we can built it and load the resulting .war file to the artifact template in the tab *Files*.

Winery				
ervice Templates Node Types	Relationship Type	Other Elements: Artifact Template	Administration	
Ubuntu-14.04-w1-wip1-C http://opentosca.org/node			elete Export -	Versions -
README LICENSE F	iles Source	Properties Property Constraints	s Documentation	XML
Durchsuchen Keine Dateien ausgewählt. You may drop the files here. The files are immediately uploaded without a	ny confirmation			ZIP
Already included Files: Add New Reload from Server Del Src/main/webapp/WEB-INF	ete Rename org opentosca	_nodetypes_Ubuntu_14_04-w1-wip1_OperatingSy	Copy all to Files	Save
vsrc/main/java/org/opentosca				^
/nodetypeimplementations	2	age org.opentosca.nodetypeimplementations		
org_opentosca_nodetypes_Ubuntu_14_0 4-w1- wip1_OperatingSystemInterface.java	4.74 KB 5 impo 6 impo 7 impo 8.61 KB 9 impo	<pre>rt java.util.HashMap; rt javax.jws.Oneway; rt javax.jws.WebMethod; rt javax.jws.WebParam; rt javax.jws.WebService; rt javax.jws.soap.SOAPBinding; rt javax.jml.bind.annotation.XmlElement;</pre>		
pom.xml 1	11 12 @web 13 publ 14 15 16 17 18 19 20 20 21 21 22	Service ic class org_opentosca_nodetypes_Ubuntu_1/ @WebMethod @SOAPBinding @Oneway public void installPackage(@WebParam(name="VMIP", targetNamespace @WebParam(name="VMPrivateKey", targetI @WebParam(name="PackageNames", targetI) { // This HashMap holds the return parar final HashMap <string,string> returnPara // TOD0: Implement your operation here</string,string>	="http://nodetypeimplement mespace="http://nodetypeimp Jamespace="http://nodetypei Jamespace="http://nodetypei meters of this operation. "ameters = new HashMap <stri< td=""><td>ations.o lementat mplement mplement</td></stri<>	ations.o lementat mplement mplement

1.4.6 Creating the Service Template

To finally model the service template, at the tab Services Templates, press Add new.

Winery			about
Service Templates No	Add a new Service Template	×	
	Name		Add new
	Python_3_ServiceTemplate		Import CSAR
Apache-Sp http://opentos	Versioning:	~	Import YAML
	Namespace		Create from Artifact
Apache-Sp http://opentos	http://opentosca.org/servicetemplates		Group by Namespace
	Template:	~	
ST FIWARE_C			
http://opentos	Cancel	Add	

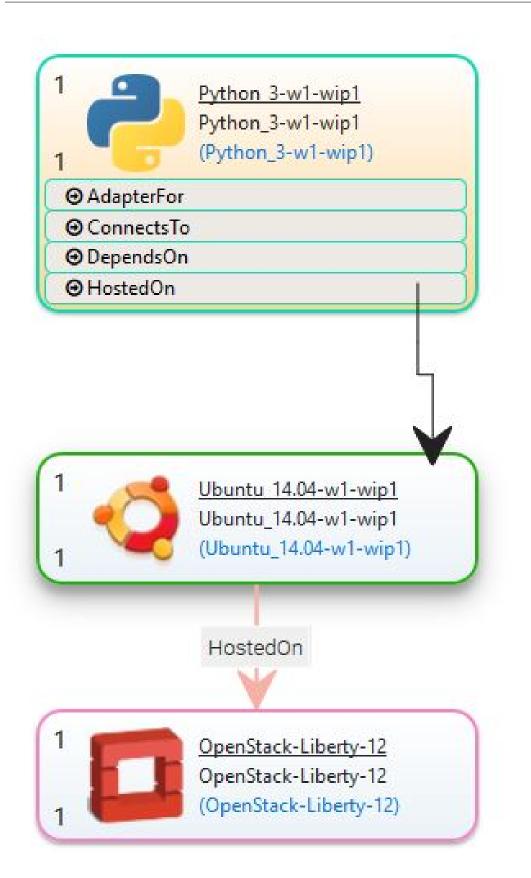
Go to tab *Topology Template* and press the button *Open Editor*.

ervice Templates	Node Types	Relationship Types	Other Elements	Administration		
	ion_3_ServiceTemplate			Dele	ete Export -	Versions -
README Constraint Check		oology Template Pla ation XML	ns Selfservice F	Portal Boundary D	efinitions	Tags
Python	3 Servi	ceTemplate	License Apache 2.0	1		
		Python3 on a Ubuntu v			frastructure	
and an out the						
	1	i yulong on a obulla i	n tuai macinie i unin	ng on an openstack in	lasti ucture.	
			intuar macinine rumm	ng on an openstate in	nashucture.	
	9 S		n tuar macinne runn	ng on an openstack in		
Propertie • OpenstackUse • OpenstackPase	PS er ssword		n tuar macinne runn	ng on an openstack in	nash ucture.	
Propertie • OpenstackUse • OpenstackPase Haftungs	es ^{ar} ssword sausschlus	s				
Propertie • OpenstackUse • OpenstackPas Haftungs Dies ist ein Forschung insbesondere bei varili sämtlicher Lizenzen g	essword saword sausschlus geprototyp und enthält inst erten oder neuen Anwendu eprüft werden. Die Haftung		n. Diese Software enthält n dere beim Produktiveinsatz iktionsausfall, Betriebsunter	röglicherweise Fehler und funk muss 1. die Funktionsfähigke brechung, entgangene Nutzun	tioniert möglicherw it geprüft und 2. die gen, Verlust von Di	e Einhaltung laten und
Propertie • OpenstackUse • OpenstackPas Haftungs Dies ist ein Forschung insbesondere bei varii sämtlicher Lizenzen g Informationen, Finanzi ausgeschlossen.	essword saword sausschlus geprototyp und enthält inst erten oder neuen Anwendu eprüft werden. Die Haftung	S besondere Beiträge von Studente ngsfällen, nicht richtig. Insbeson für entgangenen Gewinn, Produ för entgangenen Gewinn, Produ e sonstige Vermögens- und Folg	n. Diese Software enthält n dere beim Produktiveinsatz iktionsausfall, Betriebsunter	röglicherweise Fehler und funk muss 1. die Funktionsfähigke brechung, entgangene Nutzun	tioniert möglicherw it geprüft und 2. die gen, Verlust von Di	e Einhaltung laten und

In the editor, the *Palette* on the left shows the available node types, which can be drag and dropped in the modeling area.

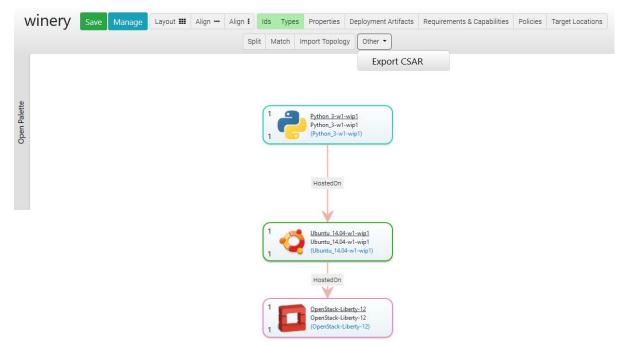
winery Save Manage	Layout 🎟	Align	Align 1	Ids Ty	pes Propert	ies D	Deployment Artifacts	Requirements & Capabilities	Policies	Target Locations
			Sp	lit Matc	n Import Top	ology	Other 🔹			
http://opentosca.org/nodetypes	^ ^									
DpenStack-Liberty-12										
Python_3-w1-wip1										
4 Ubuntu_14.04-w1-wip1										
c	>									
http://opentosca.org/nodetypes/versioned	^ û									

To model the relationship that the Python3 runtime is hosted on the Ubuntu virtual machine, click at the Python3 node template. This will show a list of possible relationship types (previously modeled). Click in the option *HostedOn* and pull the shown arrow to the Ubuntu node template area, in order to connect these node templates.



1.4.7 Exporting a Service Template Package

To export the Service Template as a CSAR package, press Other, then Export CSAR.



1.4.8 Miscellaneous Notes

Properties of a Template can be either full XML or key/value based. If key/value based, a wrapper XML element is required. Since QNames have to be unique, Winery proposes as namespace the namespace of the template appended by propertiesdefinition/winery. The name of the wrapper element is properties.

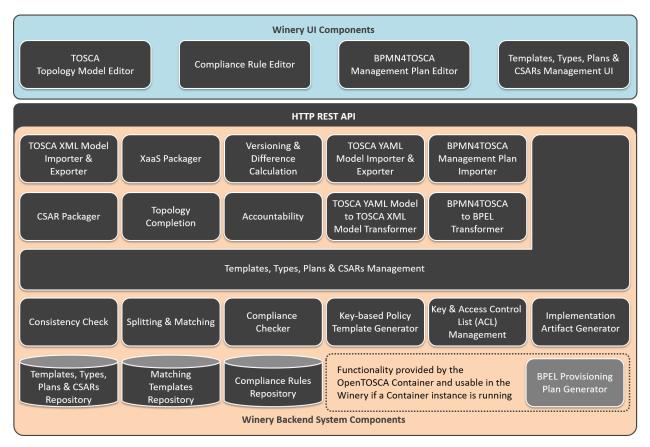
Note: Implementation hint: This is implemented in <code>PropertiesDefinitionComponent.</code> onCustomKeyValuePairSelected (TS) and org.eclipse.winery.model.tosca.TEntityType. getWinerysPropertiesDefinition (Java).

1.4.9 Uniqueness of QNames

Intentionally, a QName should be unique within the repository. We did not follow this assumption, but only require that QNames are unique within a type. That means, the repository allows {http://www.example.org}id for both a service template and a node type. We introduced DefinitionsChildId uniquely identifying a TOSCA element. Future versions might redesign the backend to use a QName as the unique key.

1.5 Component and Feature Overview

1.5.1 Components



The TOSCA modeling tool Winery mainly consists of four parts: (1) the templates, types, plans, and CSARs management, (2) the TOSCA topology model editor, (3) the BPMN4TOSCA management plan editor, and (4) the repository to store templates, types, plans, etc.

For the templates, types, plans, and CSARs management a user interface *Templates, Types, Plans & CSARs Management UI* that enables managing all TOSCA types, templates, and related artifacts is available. This includes node types, relationship types, policy types, artifact types, artifact templates, and artifacts such as virtual machine images. The *Templates, Types, Plans & CSARs Management* backend component provides functionality to access, store, or delete TOSCA elements in the *Templates, Types, Plans & CSARs Repository* which is a file system storing all available TOSCA elements.

The *TOSCA Topology Model Editor* enables the creation of service templates as directed graphs. Service templates consists of instances of node types (node templates) and instances of relationship types (relationship templates). They can be annotated with requirements and capabilities, properties, deployment artifacts, and policies. Modeled service templates can be exported based on the TOSCA XML standard using the *TOSCA XML Model Importer & Exporter* or

as YAML Model using the *TOSCA YAML Model Importer & Exporter*. Because the internal data model of the Winery is based on the XML standard the *TOSCA YAML Model to TOSCA XML Model Transformer* is required to enable the import and export as XMI as well as YAML model. The standard packaging format for service templates and all related TOSCA elements is a Cloud Service Archive (CSAR). The *CSAR Packager* backend component is responsible to package all TOSCA elements in the archive. The archive can be used by a TOSCA runtime for the deployment of the described cloud application.

The *BPMN4TOSCA Management Plan Editor* offers web-based creation of BPMN models with the TOSCA extension BPMN4TOSCA. That means, the editor supports the BPMN elements and structures required by TOSCA plans and not the full set of BPMN. The *BPMN4TOSCA Management Plan Importer* enables to load existing management plans to the Winery. Because not only BPMN but also BPEL is a common modeling language for the automated workflow execution, a *BPMN4TOSCA to BPEL Transformer* component is available to support different modeling standards. In case a running instance of the OpenTOSCA Container is available provisioning plans can be automatically generated by the *BPEL Provisioning Plan Generator*.

In addition to the described basis functionality of the TOSCA modeling tool Winery several advanced functionalities are provided:

- *Consistency Check*: This functionality enables to check whether a service template is valid according to the TOSCA XML specification. This includes the definition of used node types and properties, the QNames, and if License and README files are available. This supports the user to model valid service templates.
- *XaaS Packager*: It enables the deployment of, e.g., a web application by reusing an existing service template and replacing the deployment artifact in the specified node type with the new deployment artifact. The underlying platform or infrastructure services do not have to modeled for each application, predefined templates can be used. More information can be found here.
- *Topology Completion*: The TOSCA Topology Completion of Winery enables the user to model incomplete TOSCA Topology Templates and complete them automatically step-by-step by injecting new node templates to fulfill open requirements. More information can be found here.
- *Splitting & Matching*: The Split & Match function facilitates the redistribution of application components to target locations. For this, the application components can be annotated with target labels to indicate the desired target locations. In the *Matching Templates Repository* platform or infrastructure services can be defined as node templates or complete topology fragments for each target location. Based on the desired split, the node templates of the original service template are split according to the labels and the matching node templates or topology fragments for hosting the application's components in the target location are matched with the corresponding part of the split topology. More information can be found here.
- *Versioning & Difference Calculation*: To support version control of all TOSCA elements, including node types, artifact templates, service templates, and so on, the versioning component enables to add different versions of a TOSCA element and to release them after the development phase. Released elements can not be modified to ensure consistency of specific versions in the ecosystem. In addition, the differences between two versions can be calculated and visualized in the TOSCA Topology Model Editor.
- Accountability: In collaborative development of application deployment models in business-critical scenarios (such as data-analysis), accountability is of high importance. Thus, at CSAR export time, Winery enables to store the TOSCA meta file in a blockchain to identify the author of each exported version and whether a contained artifact is changed and by whom. Winery also stores these artifacts versions in a decentralized storage which facilitates comparing them and visualizing the provenance of a specific resource.
- Compliance Checking (Compliance Rule Editor, Compliance Checker & Compliance Rules Repository): The Topology Compliance Checking of Winery enables to describe restrictions, constraints, and requirements for Topology Templates in form of reusable topology-based Compliance Rules. These rules can be modeled using the Compliance Rule Editor and stored in the Compliance Rules Repository. Each rule consists of an Identifier and a Required Structure. If the defined identifier is contained in a topology, the required structure must be contained as well. Furthermore, the Compliance Checker of Winery can be used to ensure that a given Topology Template is compliant to the current set of Compliance Rules. More information can be found here.

- *Key-based Policy Template Generator*: This functionality allows to generate security policy templates based on keys stored in the key manager. Since a key-based security policy represents a key in a decoupled manner, the policy template only contains the details about the key, but not the key itself. Modelers can use this functionality to simplify generation of policy templates which represent respective keys.
- *Key & Access Control List (ACL) Management*: This functionality allows storing and generating symmetric keys and keypairs with self-signed certificates as well as specifying the access rules for keys for specific partner names. It allows modelers to enforce modeled security requirements at CSAR import and export times. However, this is an administrative functionality that potentially can be used for other purposes.
- *Implementation Artifact Generator*: To specify what a node type should do, the user can define an interface and the operations provided by this interface. Once the operations of a node type are defined, artifacts (e.g., shell scripts, .war files) implementing these operations need to be modeled. With the *Implementation Artifact Generator* a stub java maven project to build a .war file for a defined interface is generated automatically.
- *Grouping*: This functionality allows the grouping of node templates in the TOSCA topology model editor. It enables the possibility to model groups within a topology, e.g., to describe that a policy only applies to a certain group of node templates, but not to all node templates of a topology template.

1.5.2 Features

- Splitting Splitting functionality
- Target Allocation Select best suited cloud provider for topologies
- Topology Completion Topology completion overview
- XaaS Packager Enables reusing modeled topologies as templates for single applications
- · Compliance Checking Enables compliance checking of topology templates based on reusable rules
- Implementation Artifact Generation Shows how to generate and update an implementation artifact of type war
- · Version Management Shows how to update the version of a node template in the topology modeler
- Threat Modeling For NFV Enables threat modeling capabilities and NFV-based mitigation recommendation
- Pattern-based Deployment and Configuration Models Describes how PbDCMs can be crated and refined to executable deployment models
- Grouping Describes the usage of the grouping functionality.

1.6 Winery CLI

The Winery CLI can be used to perform a consistency check for a given repository.

- Linux: docker run -it -v \$(pwd):/root/winery-repository opentosca/winery-cli winery -v
- Windows: docker run -it -v \${PWD}:/root/winery-repository opentosca/ winery-cli winery -v

Note: You may replace \$ (pwd) or \$ { PWD } with a directory location on your Docker host system.

Currently supported CLI arguments:

-h,help	prints this help
-p,path <arg></arg>	use given path as repository path
-v,verbose	be verbose: output the checked elements

1.7 Frequently Asked Questions (FAQ)

1.7.1 Q: What is TOSCA?

A: The Topology and Orchestration Specification for Cloud Applications (TOSCA) is an OASIS standard to describe the deployment and management of applications in a portable manner. Based on standard-compliant TOSCA runtimes, such as the OpenTOSCA ecosystem, the deployment and management can be automated. For more details see our *notes on TOSCA*.

1.7.2 Q: What is a CSAR?

A: Cloud Service Archive (CSAR) is a packaging format defined by the TOSCA specification, which enables to bundle modeled TOSCA components in a self-contained manner. Besides the TOSCA elements, the executable artifacts are packed as well. In winery, you can model a service template and export it as a CSAR. This CSAR can be loaded into the OpenTOSCA container in order to deploy your application.

1.7.3 Q: How can I start the OpenTOSCA ecosystem?

A: You can start the ecosystem by simply using Docker Compose or by using installation scripts. Please refer to the OpenTOSCA getting started guide for more details.

1.7.4 Q: Is there an open repository for TOSCA types?

A: Yes! We provide a GitHub repository compatible to Winery, which contains several service templates, node types, etc. To use this repository with a Winery docker container, please refer to the corresponding configuration instructions in the user guide.

1.7.5 Q: Where can I find a quick start guide to model Node Types?

A: You can find a Winery quick start guide about modeling node types in our user guide.

1.7.6 Q: How can I export my modeled application as a CSAR?

A: Select the tab *Services Templates*. From the listed service templates, select the one you want to export. In the detailed view, press *Export* and then choose the option *CSAR (XML)*.

1.7.7 Q: My modeled Node Type got the suffix name wip what does this mean?

A: This means your node type has a *work in progress* (wip) version. That is, this node type can and might be changed. Once you are done, you can do a release of your node type. In this way, Winery will not allow changes in the (released) node type anymore.

1.7.8 Q: How can I release a Node Type?

A: Select the tab *Node Types*. From the listed node types, select the one you want to release. In the detailed view, press *Versions* and then choose the option *Release management version*.

1.7.9 Q: On Mac OS X, I can neither delete a Node Template nor a Relationship Template.

A: Select the node template (or the relationship template) and press <kbd>fn</kbd> + <kbd>backspace</kbd>.

1.7.10 Q: Where can I get more help?

A: If you need support, contact us at *opentosca@iaas.uni-stuttgart.de*.

1.7.11 Q: How can I contribute to Winery?

A: Please see the contributing guide.

CHAPTER

DEVELOPER GUIDE

This document provides an index to all development guidelines and background information of Eclipse Winery.

- Recommended Reading
- Modules Winery's module structure
- Branches How to branch
- Source Code Headers Documentation about required source code headers
- REST API How Winery's REST API works
- Encoding Information about how encoding is used in Winery
- ID System Winery's ID System
- Repository Layout Documents the layout of the repository (stored as plain text files)
- Property Handling
- Configuration and Features
- TOSCA 1.0 Notes
- IDE Setup
 - IntelliJ IDEA (recommended): config/IntelliJ IDEA
 - Eclipse: config/Eclipse
- Winery GitHub Workflow
- Setup Winery Toolchain
- Winery and Docker

2.1 Getting Started

- Clone the repository: git clone https://github.com/eclipse/winery && cd winery.
- Build Eclipse Winery: mvn clean install -DskipTests (skipping the tests for a faster build).
- Setup your IDE:
 - IntelliJ IDEA (recommended): config/IntelliJ IDEA
 - Eclipse: config/Eclipse
- Go to Eclipse Winery Toolchain for further details
- Get familiar with Winery's GitHub workflow

CHAPTER

THREE

NOTES ON TOSCA

The *Topology and Orchestration Specification for Cloud Applications (TOSCA)* is a standard defined by the OASIS organization. It defines a language to model (cloud) applications to automate their provisioning and management. Thereby, TOSCA is vendor and technology independent and aims at defining applications in a portable and interoperable manner.

In general, there are two different flavours built in to TOSCA: (i) declarative modeling and (ii) imperative modeling. While the traditional, declarative way to model an application is in the form of a *Topology Template*, i.e., a graph that describes the application's components and their relations, it also supports imperative workflows that exactly state the tasks and their order in which they have to be processed. However, since we can automatically generate the imperative workflows based on the declarative model, Winery focuses mainly on the creation of the component's types, i.e., *Node Types*, and whole applications, i.e., *Service Templates* that add additional meta-information and wrap a *Topology Template*.

For more details about the standard, go to the specifications as linked below. For more documentation about how to model an application using Winery and the OpenTOSCA ecosystem, see <../user/xml/index.rst>.

3.1 Recommended Readings

- 1. Portable Cloud Services Using TOSCA. In: IEEE Internet Computing (2012) Short overview.
- TOSCA: Portable Automated Deployment and Management of Cloud Applications. In: Advanced Web Services (2014) - Longer overview.
- 3. TOSCA Simple Profile in YAML Version 1.3 The simple profile in YAML.

See http://www.opentosca.org/sites/publications.html for a list of publications in the OpenTOSCA ecosystem.

3.2 TOSCA 1.3 YAML

- Official Specification
- Class Diagram
- PlantUML

3.3 TOSCA 1.0 XML (Deprecated)

- Official Specification
- Class Diagram
- PlantUML

3.4 Example TOSCA YAML Files

• Project RADON

3.5 Available TOSCA Implementations

• https://wiki.oasis-open.org/tosca/TOSCA-implementations

CHAPTER

ARCHITECTURAL DECISION LOG

This lists the architectural decisions for Eclipse Winery.

- ADR-0000 Use Markdown Architectural Decision Records
- ADR-0001 Use filesystem as backend
- ADR-0002 File system folder structure using type-namespace-id structure
- ADR-0003 Double Encoded URLs
- ADR-0004 OAuth with GitHub
- ADR-0005 XML editor does not enforce validation
- ADR-0006 Wrap properties in TOSCA properties element
- ADR-0007 Custom URI for lifecycle interface
- ADR-0008 No support for local git source clones
- ADR-0009 Manual serialization of SnakeYAML
- ADR-0010 TOSCA YAML deserialization using SnakeYAML
- ADR-0011 Use Builder Pattern for Model Classes
- ADR-0012 Provide Support for Custom Key-Value Properties
- ADR-0013 Routes in the Repository Angular App
- ADR-0014 Use Eclipse Orion as Editor
- ADR-0015 Offer copying files from the source to the files folder
- ADR-0016 Reflection test for TOSCA YAML builder
- ADR-0017 Modify JAX-B generated classes
- ADR-0018 Version Identifier in a Debian-like Form
- ADR-0019 Versions of TOSCA elements in the name
- ADR-0020 TOSCA Definitions contain excaly one element
- ADR-0021 Use logback for logging
- ADR-0022 tosca.model is more relaxed than the XSD
- ADR-0023 Use Maven as build tool
- ADR-0024 Use TravisCI for Continuous Integration
- ADR-0025 Use same logback-test.xml for each sub project

- ADR-0026 Store LICENSE and README.md in respective entity's root folder in a CSAR
- ADR-0027 Use dasherization for filenames
- ADR-0028 Use hardcoded namespaces for threat modeling
- ADR-0029 IPSec Algorithm Implementation
- ADR-0030 Support of multiple repositories
- ADR-0031 Reuse the pattern refinement implementation for pattern detection

The template.md contains the MADR template. More information on MADR is available at https://adr.github.io/madr/.

CHAPTER

GETTING SUPPORT FOR ECLIPSE WINERY

- In case you have concrete issues, please open an issue at https://github.com/eclipse/winery/issues.
- There is a mailing list available at https://dev.eclipse.org/mailman/listinfo/winery-dev.
- General information about Eclipse Winery is available at https://eclipse.org/winery.

CHAPTER

SIX

LICENSE

Copyright (c) 2013-2020 Contributors to the Eclipse Foundation.

See the NOTICE file(s) distributed with this work for additional information regarding copyright ownership. This program and the accompanying materials are made available under the terms of the Eclipse Public License 2.0 which is available at http://www.eclipse.org/legal/epl-2.0, or the Apache Software License 2.0 which is available at https://www.apache.org/licenses/LICENSE-2.0.

SPDX-License-Identifier: EPL-2.0 OR Apache-2.0