Sami Yangui, Mohamed Mohamed, Mohamed Sellami and Samir Tata (Telecom SudParis)
May 31 2013
Updated: July 19, 2013

Open Cloud Computing Interface - Platform

Abstract

This document provides information regarding our proposed Open Cloud Computing Interface- Platform resources extension.

Contents

1 Introduction

OCCI is a Protocol and API for all kinds of Management tasks.

As part of our work, we aim to model platform resources in the context of Cloud Computing. To do this, we opted to extend the OCCI specifications mainly dedicated for infrastructure resources modeling and defining a REST API for handling these resources [1]. These specifications are already implemented in many managers of Cloud infrastructures and consist of 3 parts:

- OCCI core that defines a meta-model for Cloud resources [2],
- OCCI infrastructure which is an instantiation of the OCCI core meta-model for infrastructure resources. OCCI infrastructure defines IaaS resources through three types (i.e. Network, Compute and Storage) and also classifies the relationship between resources in two types (StorageLink and NetworkInterface) [3],
- OCCI rendering specifications that contains multiple documents describing rendering of the OCCI Core Model [4].

In addition to these specifications, we propose in this documentan OCCI platform specifications which constitutes our proposed extension to the OCCI model for platform resources modeling.

A preliminary version of the OCCI platform specifications presented in this document was published in [5].

This document is organized as follows: Section 2 describes the main defined OCCI platform types. Section 3 details the OCCI platform resources (i.e. Database, Container, and Network). Section 4 describes platform Links between the defined OCCI platform resources. Section 5 lists a set of Mixin as examples of specific platform resources. Section 6 details an OCCI PaaS resource template (i.e. a Service-based application deployment template).

2 OCCI platform types

This section presents the main OCCI platform types of our defined extension. The platform resources types are derived directly from Resource entity at OCCI core level. The platform resource Mixins are derived from Mixin entity at OCCI core level. The interfaces which link these resources between them are derived from Link entity at OCCI core level.

Figure 1 details the overview of the different OCCI platform types and the connection of these types with the OCCI core entities.

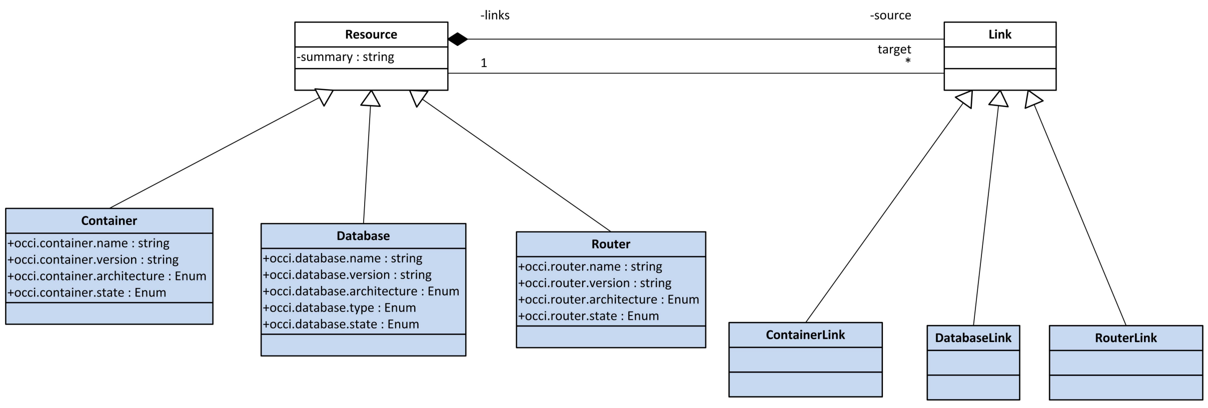
Resource
-summary : string
-links
1
-source
target
*
Link
Container
+occi.container.name : string
+occi.container.version : string
+occi.container.architecture : Enum
+occi.container.state : Enum
Database
+occi.database.name : string
+occi.database.version : string
+occi.database.architecture : Enum
+occi.database.type : Enum
+occi.database.state : Enum
Router
+occi.router.name : string
+occi.router.version : string
+occi.router.architecture : Enum
+occi.router.state : Enum
ContainerLink
DatabaseLink
RouterLink

Figure 1. Overview of the defined OCCI types.

The main OCCI platform types defined are:

- Database which are data store resources for platform applications which process persistent Data. Database nodes can be relational database (e.g. MySQL, PostgreSQL, etc.) or non-relational database (e.g. MongoDB, CouchDB, etc.),
- Container which are engines to host and run applications (e.g. Apache Axis container, Bonita, IBM WebSphere, etc.),
- Router which are resources that provide protocols and messages format transformation and routing (e.g. ESBPetals router, Apache Synapse, etc.).

We also define relations between platform resources using different links provided for this purpose:

- ContainerLink to connect Container resources,
- RouterLink to connect Router resources,
- DatabaseLink to connect a Container resource to a Database resource.

In addition to that, a set of platform resource Mixins can be defined if needed through this extension to support specific platform resource features (e.g authentication platform features, logging platform features, etc.).

Table 1 describes the Kind instances defined for each one of the platform Resource or Link sub-types. For information on extending these types, please refer to the OCCI Core Model document [2].

Table 1. The kind instances defined for the platform subtypes of Resources, Links and related Mixins.

Term	Scheme	Title	RelatedKind
Database	$< schema > /paas\#$	Database Resource	$< schema > /core\#resource$
Container	$< schema > /paas\#$	Container Resource	$< schema > /core\#resource$
Router	$< schema > /paas\#$	Router Resource	$< schema > /core\#resource$
DatabaseLink	$< schema > /paas\#$	Database Link	$< schema > /core\#link$
ContainerLink	$< schema > /paas\#$	Container Link	$< schema > /core\#link$
RouterLink	$< schema > /paas\#$	Router Link	$< schema > /core\#link$

3 OCCI platform resources

This section details attributes, actions and defined states for each one of the defined OCCI platform resource types.

3.1 Database resource

The Database resource type represents storage resources which can be provisioned by a platform provider for applications which process persistent Data (e.g. MySQL[1], Apache CouchDB[2], etc.). The Database type inherits the Resource base type defined in the OCCI Core Model [2].

Table 2 describes the attributes describing the Database type through its Kind instance (Database scheme). These attributes are exposed by all Database type instances.

Table 3 describes the actions applicable to the Database type by its Kind instance. Every action in the table is identified by a Category instance using a $/database/action\#$ categorization scheme. 'Action Term' refers to the term of the Action's Category identifier. These actions are exposed by all Database type instances of an OCCI implementation.

Figure 2 illustrates the state diagram for a Database instance.

[1]`http://www.mysql.com/`
[2]`http://couchdb.apache.org/`

Table 2. Attributes describing the Database type.

Attribute	Type	Multi-plicity	Mutability	Description
occi.database.name	String	0...1	Mutable	Name of the instance.
occi.database.type	Enum {relational, keyValue, document, graph}	0...1	Mutable	Scheme type of the instance.
occi.database.architecture	Enum {x86, x64}	0...1	Mutable	CPU architecture of the instance.
occi.database.version	String	0...1	Mutable	Version Label of the instance.
occi.environment.state	Enum {available, unavailable}	1	Immutable	Current state of the instance.

Table 3. Actions applicable to instances of the Database type.

Action Term	Target state	Attributes
StartDB	Available	database credentials
StopDB	Unavailable	–
RestartDB	Available (via stop and start chain)	database credentials
BackupDB	None	–

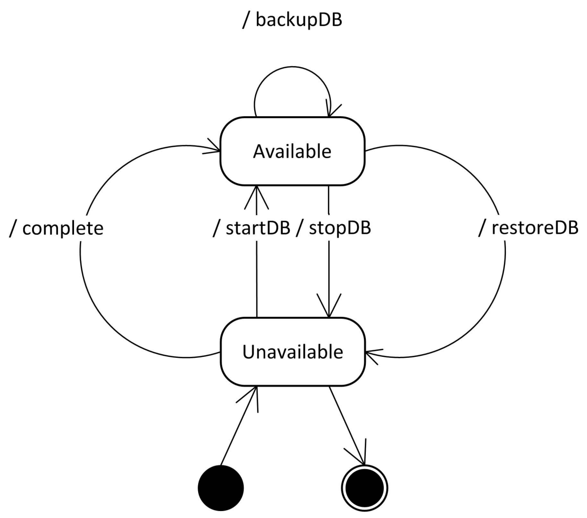

Figure 2. State Diagram for a Database instance.

3.2 Container resource

The Container resource type represents service containers and engines provisioned by platform providers to host and run applications (e.g. Apache Axis[3], Oracle GlassFish Server[4], etc.).

Table 4 describes the attributes describing a Container type through its Kind instance (Container scheme). These attributes are exposed by all Container type instances.

Table 4. Attributes defined for the Container type.

Attribute	Type	Multi-plicity	Mutability	Description
occi.container.name	String	0...1	Mutable	Name of the instance.
occi.container.version	String	0...1	Mutable	Version Label of the instance.
occi.container.architecture	Enum {x86, x64}	0...1	Mutable	CPU architecture of the instance.
occi.container.state	Enum {available, restarting, un-available}	1	Immutable	Current state of the instance.

[3]http://axis.apache.org/axis2/java/core/
[4]https://glassfish.java.net/

Table 5 describes the actions applicable to the Container type by its Kind instance. Every action in the table is identified by a Category instance using a $/container/action\#$ categorization scheme. 'Action Term' refers to the term of the Action's Category identifier. These actions are exposed by all Container type instances of an OCCI implementation.

Table 5. Actions applicable to instances of the Container type.

Action Term	Target state	Attributes
StartContainer	Available	–
RestartContainer	None	–
StopContainer	Unavailable (	–

Figure 3 illustrates the state diagram for a Container instance.

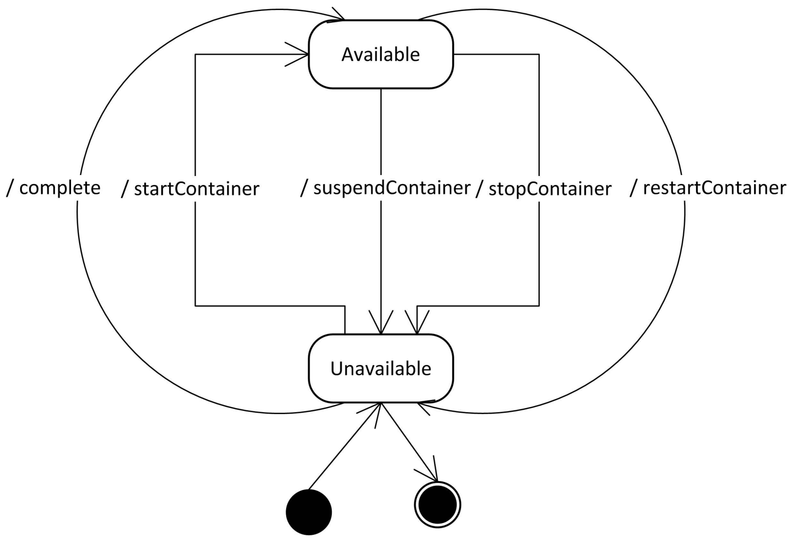

Figure 3. State Diagram for a Container instance.

3.3 Router resource

The router resource type represents message format transformation and routing systems provided by platform providers to route and deliver messages between containers. Router entities are useful where deployed applications are multi-tenant and/or service-based and requires several (may be heterogeneous) containers.

Table 6 describes the attributes describing the Router type through its Kind instance (Router scheme). These attributes are exposed by all Router type instances.

Table 6. Attributes describing the Router type.

Attribute	Type	Multi-plicity	Mutability	Description
occi.router.name	String	0...1	Mutable	Name of the instance.
occi.router.version	String	0...1	Mutable	Version Label of the instance.
occi.router.architecture	Enum {x86, x64}	0...1	Mutable	CPU architecture of the instance.
occi.router.state	Enum {available, configuring, Inactive}	1	Immutable	Current state of the instance.

Table 7 describes the actions applicable to the Router type by its Kind instance. Every action in the table is identified by a Category instance using a $/router/action\#$ categorization scheme. 'Action Term' refers to

Table 7. Actions applicable to instances of the Router type.

Action Term	Target state	Attributes
EnableRouter	Active	–
ConfigureRouter	None	Configuration parameters
DisableRouter	Inactive (	–

the term of the Action's Category identifier. These actions are exposed by all Router type instances of an OCCI implementation.

Figure 4 illustrates the state diagram for a Router instance.

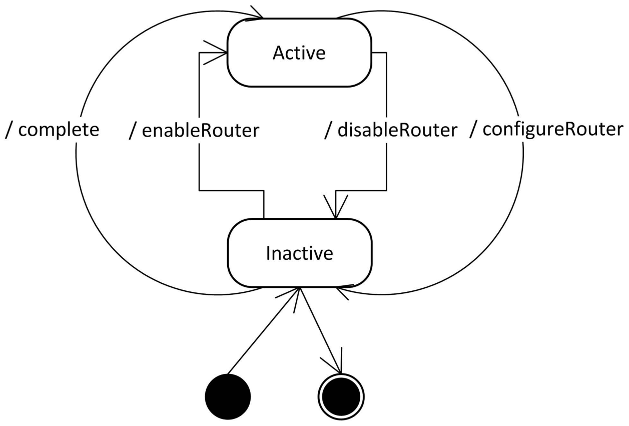

Figure 4. State Diagram for a Router instance.

4 OCCI platform Links

This section describes how defined platform resources can be linked in order to connect platform components. This is accomplished by extending the OCCI core Model Link base type.

4.1 DatabaseLink

The DatabaseLink type represents a binding between a Container and a Database resource. This enables a Database instance to be attached to a Container instance (See Figure 5).

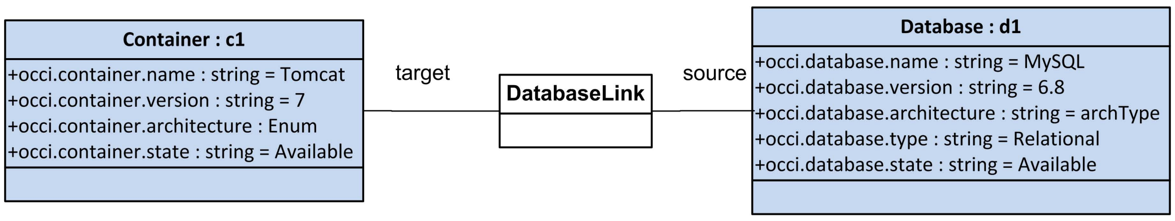
Container : c1
+occi.container.name : string = Tomcat
+occi.container.version : string = 7
+occi.container.architecture : Enum
+occi.container.state : string = Available
target
DatabaseLink
source
Database : d1
+occi.database.name : string = MySQL
+occi.database.version : string = 6.8
+occi.database.architecture : string = archType
+occi.database.type : string = Relational
+occi.database.state : string = Available

Figure 5. DatabaseLink type: A binding between Container and Database resources.

Table 8 defines the actions applicable to a DatabaseLink.

As examples of DatabasesLink, we can cite:

- .NET/Connector: To connect a .NET container to a MySQL instance.
- Mongo+Hadoop Connector : To connect a Hadoop server to a MangoDB instance.

Table 8. Actions applicable to instances of a DatabaseLink.

Action	Attributes	Description
bind	Source, target	Bind a Container source with a target Database instance.

4.2 RouterLink

The RouterLink represents the link through which it is possible to connect to a router instance (See Figure 6).

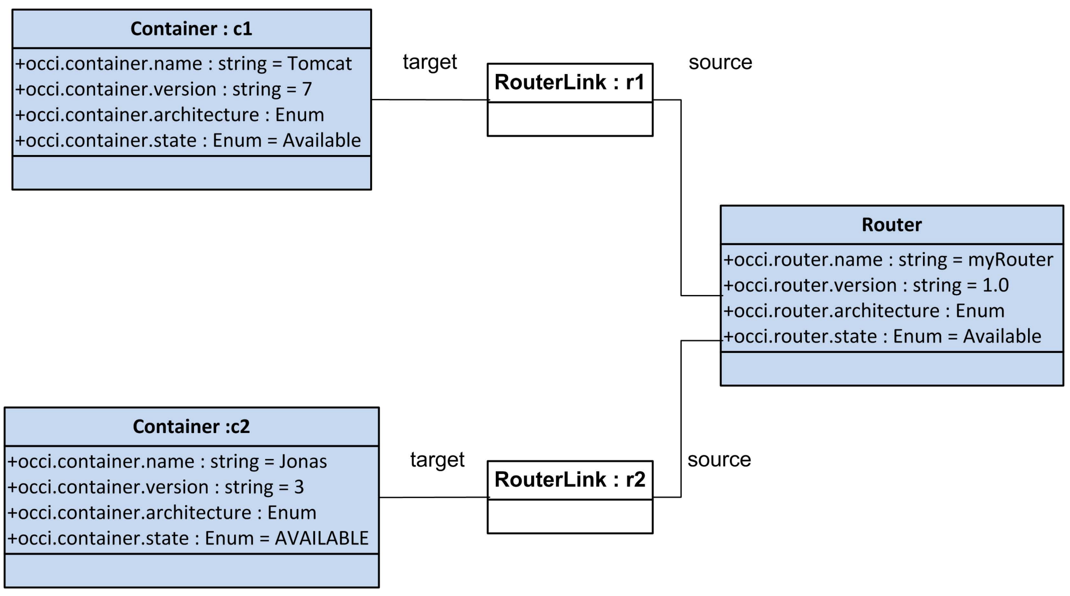
Container : c1
+occi.container.name : string = Tomcat
+occi.container.version : string = 7
+occi.container.architecture : Enum
+occi.container.state : Enum = Available
target
RouterLink : r1
source
Router
+occi.router.name : string = myRouter
+occi.router.version : string = 1.0
+occi.router.architecture : Enum
+occi.router.state : Enum = Available
Container :c2
+occi.container.name : string = Jonas
+occi.container.version : string = 3
+occi.container.architecture : Enum
+occi.container.state : Enum = AVAILABLE
target
RouterLink : r2
source

Figure 6. RouterLink.

4.3 ContainerLink

The ContainerLink represents the link through which it is possible to connect multiple container instances (See Figure 7).

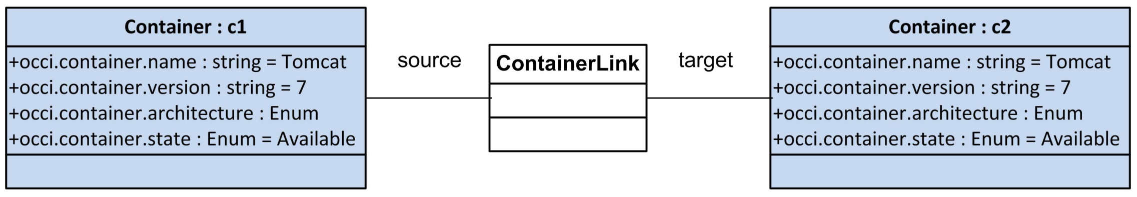
Container : c1
+occi.container.name : string = Tomcat
+occi.container.version : string = 7
+occi.container.architecture : Enum
+occi.container.state : Enum = Available
source
ContainerLink
target
Container : c2
+occi.container.name : string = Tomcat
+occi.container.version : string = 7
+occi.container.architecture : Enum
+occi.container.state : Enum = Available

Figure 7. ContainerLink.

5 OCCI platform mixins

This section lists some examples of idetified Mixins. These mixins are defined to support specific features and operations offered by some platform resources and can not be described by the main platform resources intoduced in Section 3

5.0.1 Micro-container Mixin

In order to support our defined service micro-container capabilities [6, 7] (e.g. service container mobility [8], service container monitoring [9], etc.), a Micro-container Mixin is defined.

Service micro-containers are specific types of containers [7]. They are generated dynamically at deployment type around the hosted service and are only composed of the necessary modules for the deployed service, no more, no less.

Table 9 defines the attributes describing the micro-container Mixin. A Container instance associated to the micro-container Mixin instance implements all these attributes.

Table 9. Attributes describing the MicroContainer Mixin.

Attribute	Type	Multi-plicity	Mutability	Description
MC- FileName	String	1	Immutable	The MC executable File name (e.g. A jar file name).
requirements	Set of String	1	Immutable	List of requirements needed to be run the MC (e.g. JRE version, parameters to provide JRE at start time, etc.).
state	Enum {available, migrating, restarting, unavailable}	1	Immutable	Current state of the Mixin instance.

Table 10 describes the actions defined for a micro-container Mixin.

Table 10. Actions applicable to instances of the micro-container type.

Action Term	Target state	Attributes
MonitorMC	None	Monitoring type
MigrateMC	Available	Target host

Figure 8 illustrates the state diagram for a micro-container Mixin instance.

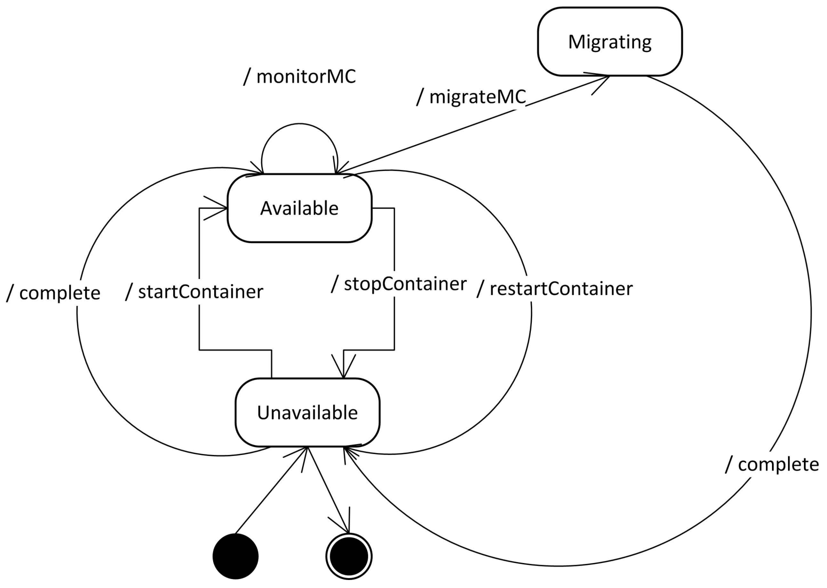

Figure 8. State Diagram for a Micro Container Mixin instance.

5.0.2 WSO2 ESB Mixin

WSO2 ESB is an Open Source Enterprise Service Bus (ESB)[5]. It allows users to configure message routing, virtualization, intermediation, transformation, logging, task scheduling, load balancing, failover routing and event brokering.

WSO2 ESB design is extensible and allows therefore integrating new modules if needed from a remote P2 repository (e.g. installing the Carbon UI Feature) [10]. To support all these features, we can define a WSO2 Mixin. Table 11 describes the actions defined for a WSO2 Mixin [10, 11].

Table 11. Actions applicable to instances of the WSO2 Mixin.

Action Term	Target state	Attributes
installModule	Available	Module name, module version, module provider

Figure 9 illustrates the state diagram for a WSO2 Mixin instance.

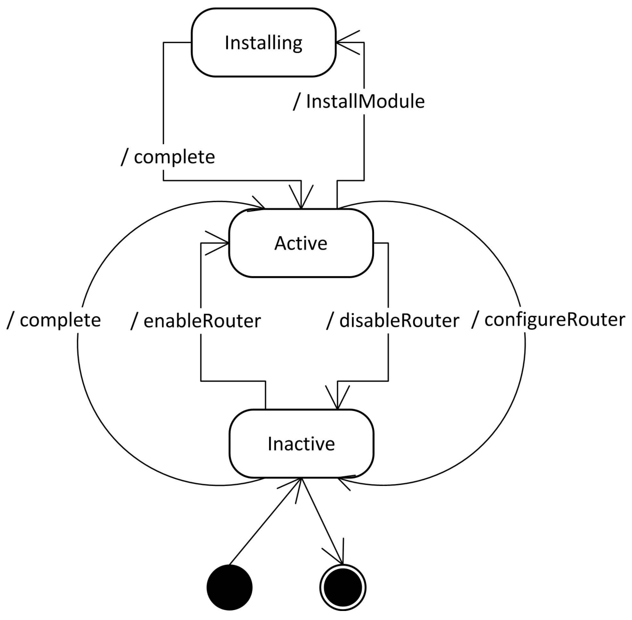

Figure 9. State Diagram for a WSO2 Mixin instance.

5.0.3 Other Mixins

Platform providers proposes technical services and features (e.g. authentication service, logging service, metering service, monitoring service, etc.) that can be supported by our extension and modeled as Mixins. In addition to that, some platform providers hosts paid applications in marketplaces which can be also modeled as Mixins.

6 Service-based application deployment template: a platform resources template

Platform templates allow users to quickly and conveniently apply predefined configurations to the OCCI platform defined types. Figure 10 describes a defined OCCI resource template for service-based applications deployment and its derived mixins.

[5]http://wso2.org/project/esb/java/3.0.0/docs/index.html

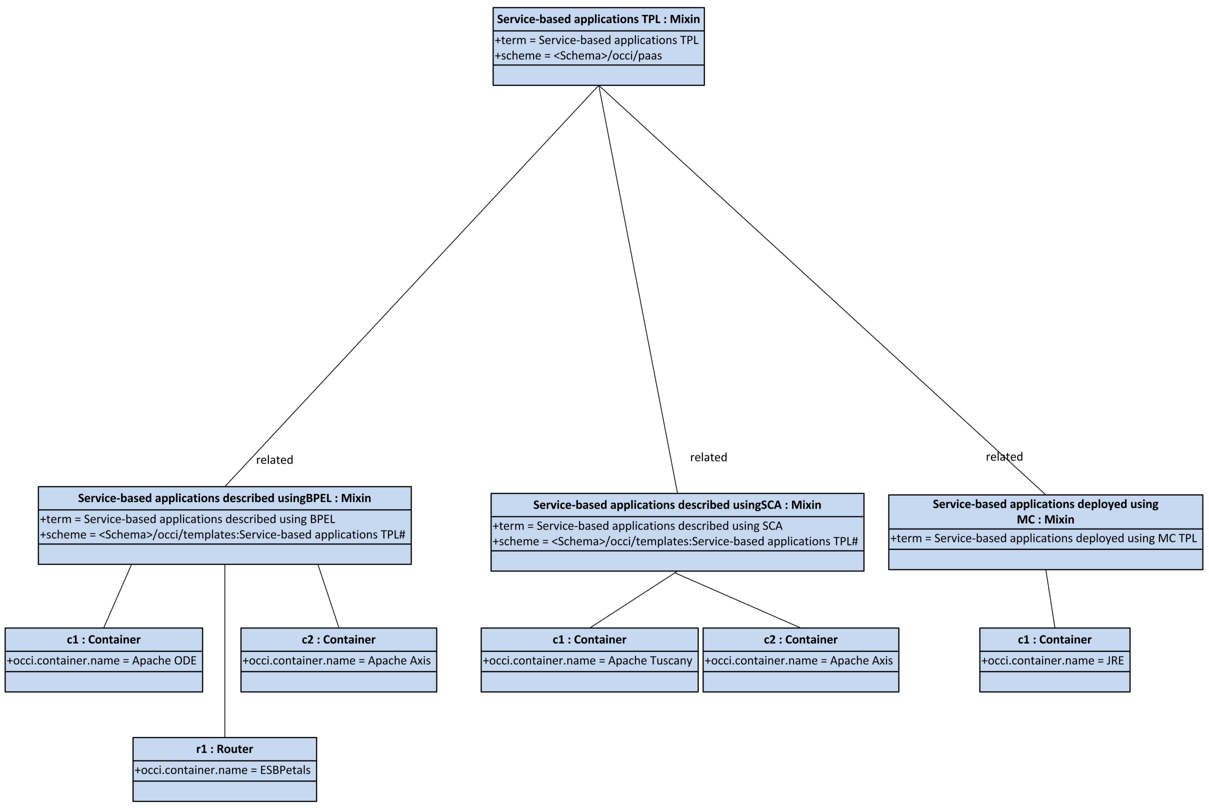
Service-based applications TPL : Mixin
+term = Service-based applications TPL
+scheme = <Schema>/occi/paas
related
related
related
Service-based applications described usingBPEL : Mixin
+term = Service-based applications described using BPEL
+scheme = <Schema>/occi/templates:Service-based applications TPL#
Service-based applications described usingSCA : Mixin
+term = Service-based applications described using SCA
+scheme = <Schema>/occi/templates:Service-based applications TPL#
Service-based applications deployed using MC : Mixin
+term = Service-based applications deployed using MC TPL
c1 : Container
+occi.container.name = Apache ODE
c2 : Container
+occi.container.name = Apache Axis
r1 : Router
+occi.container.name = ESBPetals
c1 : Container
+occi.container.name = Apache Tuscany
c2 : Container
+occi.container.name = Apache Axis
c1 : Container
+occi.container.name = JRE

Figure 10. platform resources instances for Service-based applications deployment template

7 Contributors

Name	Affiliation	Contact

References

[1] "Open cloud computing interface - occi," http://occiwg.org/, 2013.

[2] R. Nyrén, A. Edmonds, A. Papaspyrou, and T. Metsch, "Open Cloud Computing Interface – Core," GFD-P-R.183, April 2011. [Online]. Available: {http://ogf.org/documents/GFD.183.pdf}

[3] T. Metsch and A. Edmonds, "Open Cloud Computing Interface – Infrastructure," GFD-P-R.184, April 2011. [Online]. Available: {http://ogf.org/documents/GFD.184.pdf}

[4] ——, "Open Cloud Computing Interface – HTTP Rendering," GFD-P-R.185, April 2011. [Online]. Available: {http://ogf.org/documents/GFD.185.pdf}

[5] S. Yangui and S. Tata, "CloudServ: PaaS Resources Provisioning for Service-Based Applications," 27th IEEE International Conference on Advanced Information Networking and Applications, AINA 2013, March 2013. [Online]. Available: {http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6512431}

[6] M. Mohamed, S. Yangui, S. Moalla, and S. Tata, "Web service micro-container for service-based applications in cloud environments," in *WETICE*, S. Reddy and S. Tata, Eds. IEEE Computer Society, 2011, pp. 61–66.

[7] S. Yangui, M. Mohamed, S. Tata, and S. Moalla, "Scalable service containers," in *CloudCom*, C. Lambrinoudakis, P. Rizomiliotis, and T. W. Wlodarczyk, Eds. IEEE, 2011, pp. 348–356.

[8] A. Omezzine, S. Yangui, N. Bellamine, and S. Tata, "Mobile service micro-containers for cloud environments," in *WETICE*, S. Reddy and K. Drira, Eds. IEEE Computer Society, 2012, pp. 154–160.

[9] M. Mohamed, D. Belaïd, and S. Tata, "How to provide monitoring facilities to services when they are deployed in the cloud?" in *CLOSER*, F. Leymann, I. Ivanov, M. van Sinderen, and T. Shan, Eds. SciTePress, 2012, pp. 258–263.

[10] "Wso2 admin guide," http://wso2.org/project/esb/java/3.0.0/docs/admin_guide.html#P2, 2013.

[11] "Extending wso2 enterprise service bus (esb)," http://wso2.org/project/esb/java/3.0.0/docs/extensions_guide.html, 2013.