Resource Description in Large Scale Heterogeneous Resource Federations

Sebastian Wahle[1], Christos Tranoris[2], Shane Fox[3], and Thomas Magedanz[1]

[1] Fraunhofer FOKUS, Germany
{sebastian.wahle,thomas.magedanz}@fokus.fraunhofer.de
[2] University of Patras, Greece
tranoris@ece.upatras.gr
[3] Telecommunications Software & Systems Group, WIT, Ireland
sfox@tssg.org

Abstract. Resource Federations aim at providing access to information and communication technology (ICT) resources across the boundaries of administrative domains. This is of interest today as modern societies are concerned about ICT infrastructure energy consumption and need to improve the way ICT resources are provisioned and maintained. This paper describes a concept and prototype implementation for resource federations to overcome resource and implementation heterogeneity in order to allow easy resource provisioning and control. This is achieved by defining a Resource Adaptor Description Language (RADL) that allows Domain Managers and higher layer orchestration logic to control heterogeneous resources abstracting from programming languages and implementation paradigms. The prototype has been evaluated by deploying RADL within the Panlab federation. The paper summarizes our experiences and outlines the most important results.

Keywords: Resource Adaptor Description Language, RADL, Teagle, Panlab, Resource Federation, FIRE, Testing, Testbed, Future Internet.

1 Introduction

In resource federations, several organizations commit resources to a resource pool in order to implement a common service. Therefore, resource federation is a concept to allow resource sharing beyond the boundaries of administrative and organizational domains. This mechanism can follow a recursive model. Several research initiatives worldwide currently address this challenge for a number of reasons:

- Modern societies are concerned about the ICT industry energy consumption. Sharing and re-using infrastructure and services across organizations is expected to reduce the overall energy consumption, as well as over provisioning to ensure high availability of services.
- The pace of network convergence and technology evolution has dramatically decreased infrastructure lifetime – the time an infrastructure remains at the technology's cutting edge – making investments in expensive isolated and specialized infrastructures more risky than they were already. [1]

- Large scale ICT and network research experiments carried out using live networks and production systems require federated infrastructural resources to increase the scale and realism of experiments [2].
- The above mentioned points apply in particular to complex cross-layer and cross-technology infrastructures such as Future Internet (FI) research testbeds.

There is no generally accepted definition of FI. In the context of this work, FI can be defined as a large scale socio-technical system, comprising of Internet-accessible infrastructure, information and services, coupled with the physical environment and human behaviour. [3] For FI research testbeds the following applies:

- Federation enables access to additional resources increasing the scale of potential experiments. [3]
- Federation enables access to resources with unique properties to enrich experiments. [3]
- Combining resources from different communities promotes the collaboration between these and the related research groups (e.g. Telco and Internet). [3]
- A collection of testbeds that share or feature similar properties or technologies might eventually evolve into the backbone of the Future Internet itself.

This has led to numerous research programs in the field of new Internet architectures as well as suitable experimental platforms to support large scale experiments. Examples are the NSF programs GENI [4] and FIND [5] as well as the European FIRE initiative [6], [7]. In Asia similar programs have been launched such as AKARI [8] in Japan. To support architectural research experiments, several experimental facility projects have been launched such as TIED [9], PlanetLab [10], ProtoGENI [11], ORCA [12], and ORBIT [13] on the GENI side. In FIRE, several projects are contributing to the experimental facility such as Onelab2 [14], Federica [15], and PII [1],[16],[17]. Joint Asian activities are carried out under the APAN (Asia-Pacific Advanced Network) [18] initiative, the Asia Future Internet Forum (AsiaFI) [19], as well as PlanetLab CJK (China, Japan, Korea), which is a joint PlanetLab cooperation by China, Japan, and Korea [20]. An in-depth discussion and comparison between the different control framework approaches for experimental facilities has been published earlier [21].

Most of the initiatives and projects mentioned above are currently designing and implementing federation mechanisms and procedures with specific use cases and application areas in mind. Many of the aspects that need to be dealt with in this context are not new and have been achieved in different applications domains in the past. For example, computing power federation has been tackled in the Grid domain. Another example is identity federation, which has been solved for roaming in Telco networks. [3]

However, federating arbitrary resources across multiple administrative domains and on multiple federation levels, involves so many technical, operational, and legal issues that it can be considered a valid research field with many yet unsolved issues. In order to realize the vision of fully federated information and communication technology resources that can be used transparently and seamlessly, the following fields have to be addressed: resource description, resource registration, resource access control, service level agreements, resource usage policies, resource management, resource life cycle, operational procedures, legal frameworks, provider/user incentives, business frameworks, market platforms, etc. [3].

Furthermore, although many of the above listed issues have been addressed and widely discussed for single domains, additional constraints arise for multi level federations where administrative domains allow resource usage beyond the first layer of abstraction. For example, a university might establish a resource federation where different departments adhere to a centralized resource control/management instance, resource description model, operational procedures, etc. and commit resources to a university-wide resource pool. The university might now join a nationwide initiative (e.g. GENI) where several universities with similar resource control/management schemes agree to federate. This federation is then essentially a federation of federations. The next level is still imaginable: a federation of nationwide federations (e.g. GENI and FIRE agree to federate). This is basically a recursive model that can be investigated at any meaningful granularity [3].

The paper is structured as follows: section 2 covers our approach to resource federation in terms of a federation framework and resource modeling/description. Section 3 introduces the Resource Adaptor Description Language, while section 4 outlines concrete usage examples. Section 5 concludes the paper.

2 Federation Framework and Resource Description

We have developed a Resource Federation Model [1] and an according prototype implementation [1], [23] that allows sharing resources beyond domain boundaries. As this has been discussed in previous publications, as cited above, we will only very briefly summarize this for the convenience of the reader.

An important design criterion was that the resources to be shared should be highly heterogeneous in nature. Resources that are currently supported range from general-purpose virtual and physical machines, to Cloud Computing resources, services, and specialized devices. Via dedicated Resource Adaptors (RA, similar to device drivers) that plug into Domain Managers (e.g. the Panlab Testbed Manager, PTM [23]), arbitrary resources can be controlled making it a generic control framework that strongly supports federation. Experimenters can browse through the resource registry content and can select, configure, deploy, and access booked resources in order to execute network and application layer experiments.

Fig. 1 shows this concept of distributed heterogeneous resources that are offered by distributed testbeds. The resources are described according to a common model. This allows for sophisticated resource management across the boundaries of organizational domains. This concept allows us to provide a large pool of federated resources that can be used in any meaningful combination.

It is planned to federate our resources beyond the Panlab federation with similar approaches in the United States and Asia. However, this results in a new set of challenges and requirements, as federating across federations requires all layers to interoperate. Also, issues like federated identity management, federated resource control frameworks, domain specific resource and policy descriptions, etc. need to be overcome. In Panlab, some of these issues have been tackled by agreeing upon and specifying critical functionality such as resource description and control framework interfaces. It remains to be seen if sufficient demand for global heterogeneous resource federations will be observed and if both technical and operational/legal solutions can be found to address this challenge.

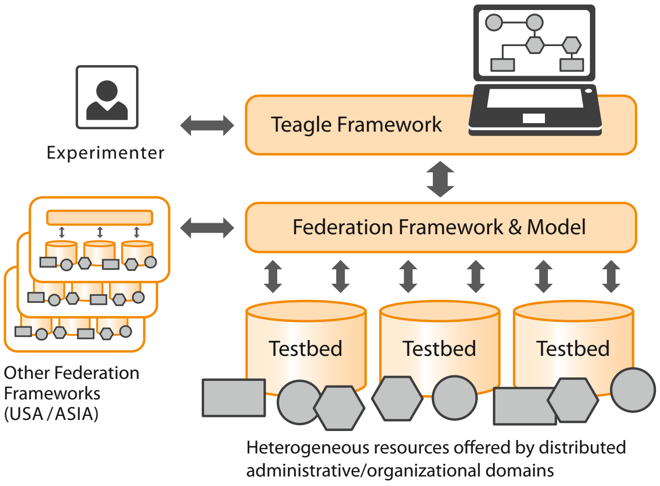

Experimenter
Teagle Framework
Federation Framework & Model
Other Federation Frameworks (USA / ASIA)
Testbed
Testbed
Testbed
Heterogeneous resources offered by distributed administrative/organizational domains

Fig. 1. Overall resource federation framework overview

Fig. 2 shows how resources are controlled inside one Panlab domain. Virtual groupings of resources can span the border of domains. The central Teagle framework allows the configuration, orchestration, and reservation of such virtual resource assemblies relying on a common resource description model and central registry.

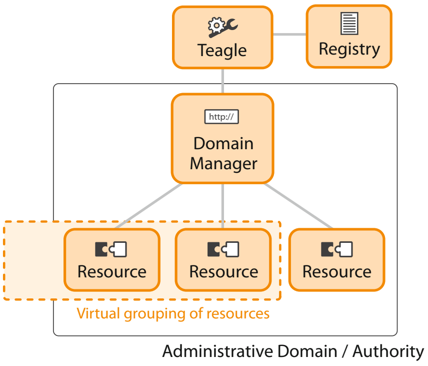

Teagle
Registry
http://
Domain Manager
Resource
Resource
Resource
Virtual grouping of resources
Administrative Domain / Authority

Fig. 2. Control framework model

Teagle in itself is a rather complex framework and compromises several entities such as a request processor, an orchestration engine, a resource repository, as well as a customizable graphical user interface. All this has been described in [23].

In the following, we will focus on the resource registry and resource description model used in Panlab [17] that allows assembling and managing virtual groupings of distributed resources. In Panlab, a virtual resource grouping is a specific testbed requested by a Panlab customer, we call this a VCT (virtual customer testbed).

Given that the federation system needs to deal with a great number of highly heterogeneous resources, the model used to structure and describe the resources needed to be extensible. An existing information model was used as the basis to represent characteristics of the resources and their relationships in a common form independent of a specific repository implementation. Resources can be modeled as concrete physical entities such as a physical machine or abstract, logical resources such as an administrative domain.

The DEN-ng information model [22] that is rooted in the area of network management and particularly autonomic networking was taken as a starting point for the modeling work. It allows the description of resources as managed entities, their life cycle, as well as associated policies to be modeled. In terms of DEN-ng, resource entities provide a service and have a certain configuration attached that can be defined and altered using the federation tools that are exposed to the experimenter via Teagle [1].

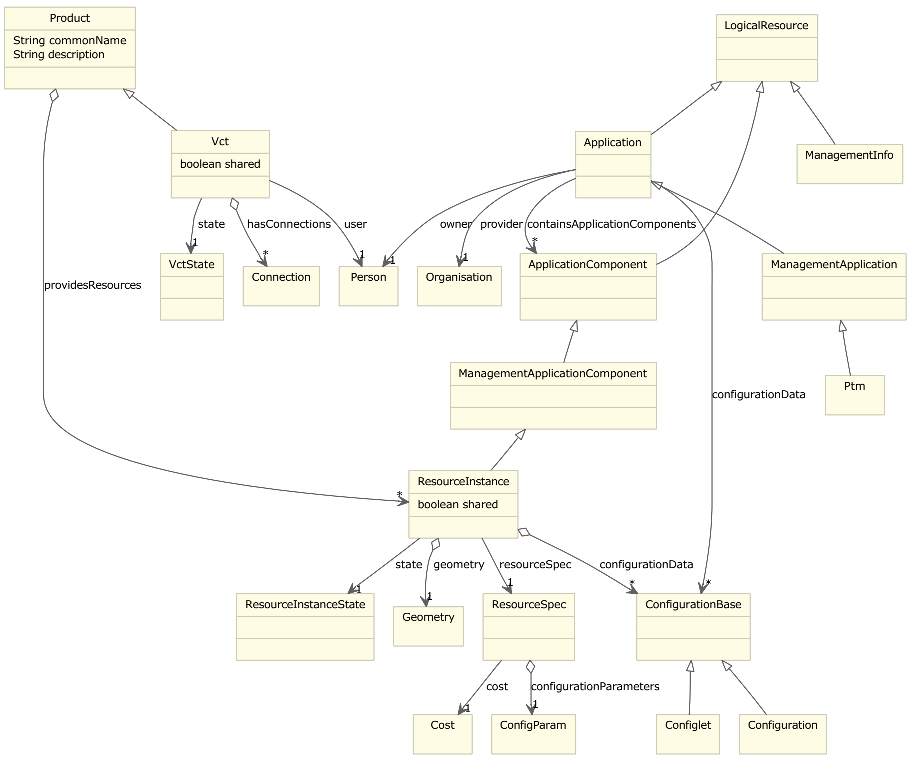


Fig. 3. Excerpt from the information model, showing the LogicalResource branch

Resources can exist as Physical- or Logical-Resource (see Fig. 3) where resource providers can define a list of resource instances as specific subtypes based on the model to represent their federation offerings. The repository implementation has been realized as a number of applications running as contexts on an application server. Each application has its own data storage facility and exposes a HTTP-based RESTful interface with a number of REST (REpresentation State Transfer) resources. The repository only deals with storage and retrieval of data on behalf of client applications. Other tasks are carried out by specially designed applications using the repository for data storage. This allows the set of tools that collectively represent the Teagle framework, to develop independently of the repository but to rely on a common data model.

The HTTP-based RESTful interface exposed by the implementation allows access to data via the HTTP standard methods of GET, POST, PUT and DELETE. RESTful calls on resources are implemented in the form of URL mappings to controller methods. As an example, the response for the GET request for a VCT (id=20) is listed below. This shows the configuration for a given VCT containing several resource instances and lists the connections between the resource instances.

```
<vct id="20">
  <shared>false</shared>
  <hasConnections>
    <connection id="3020"/>
    <connection id="3021"/>
    <connection id="3022"/>
  </hasConnections>
  <commonName>My_VCT</commonName>
  <hasBookings/>
  <description> My_VCT </description>
  <state id="3"/>
  <providesResources>
    <resourceInstance id="5169"/>
    <resourceInstance id="5175"/>
    <resourceInstance id="5539"/>
    </providesResources>
 <user id="5"/>
</vct>
```

Each resource instance has other data sets defining the configuration for the resource instance. These data sets are called configlets as shown below.

```
<resourceInstance id="5539">
  <shared>false</shared>
  <resourceSpec id="3188"/>
  <commonName>VideoLan-16</commonName>
  <configurationData>
    <configlet id="7506"/>
    <configlet id="7500"/>
    <configlet id="7503"/>
    <configlet id="7499"/>
  </configurationData>
  <description>singleton VideoLan-16</description>
  <state id="9"/>
  <geometry id="694"/>
  <availability/>
</resourceInstance>
```

Resource instances are derived from resource types. For every resource type there can be many instances with different configurations that can be part of numerous VCTs. An example for a resource instance is a specific deployed virtual machine that is of type virtual machine and can be configured with a certain amount of memory and a number of central processing units (CPUs). Further details of the Panlab repository, its information model and implementation can be found in [29]. In the following section, a Resource Adapter Description Language (RADL) is introduced that shall help developers implement resource adaptors. It is critical for the Panlab federation to maintain a rich collection of resources that can be offered in order to attract many experiments to be run on the federated facility. Therefore, making it easy to offer resources via Panlab is a main concern. Hence, the development of RADL.

3 Resource Adaptor Description Language

The Resource Adapter Description Language (RADL) is a concrete textual syntax for describing a Resource Adapter (RA, see section 2) based on an abstract syntax defined in a meta-model. RADL is an attempt to describe an RA in a way that decouples it from the underlying implementation code. RADL's textual syntax aims to simplify the description of an RA rather than having to write code in Java or other target languages. We anticipate that one can define an RA in RADL without even knowing the target language. This description could be also used to publish the RA definition to the repository for example defining the parameters as part of a `ConfigParam` (see Fig. 3).

RADL is useful in cases where there is a need to configure a resource that offers an API for configuration as illustrated in Fig. 4. The user can configure the resource through some Configuration Parameters. The RA "wraps" the parameters and together with the Binding Parameters, the RA can configure the resource. A Binding Parameter is a variable that is assigned locally by the resource provider, e.g. a local IP address.

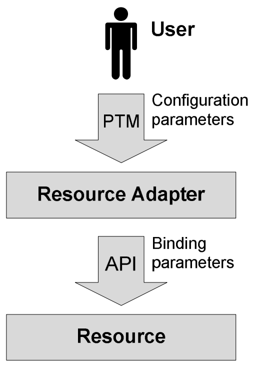


Fig. 4. A Resource Adaptor configures a resource through an API

The abstract syntax of the language, the RADL meta-model, is defined in Ecore: a variant of OMG's MOF [24] that has been defined in the Eclipse Modeling Framework [25] and is more or less aligned with OMG's Essential MOF (EMOF).

Part of the meta-model is illustrated in Fig. 5, where the Resource Adapter is an aggregation of some parameters, particularly the `BindingParam` and `ConfigurationParam`. The class Protocol wraps the concept of the API configuration.

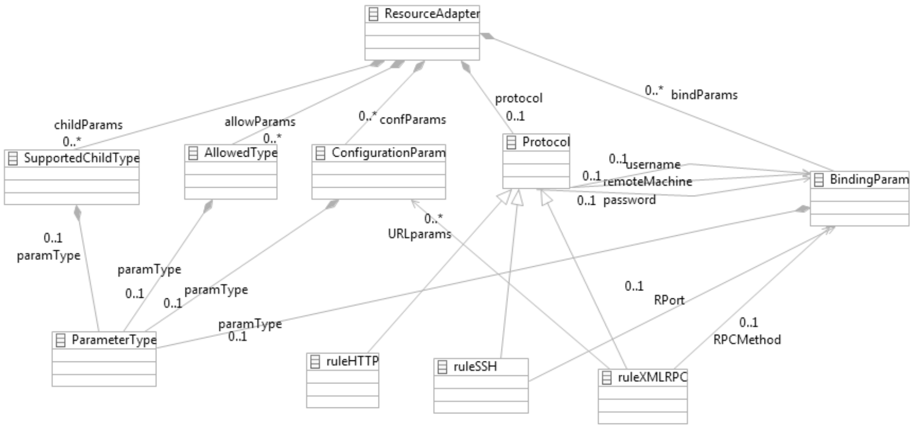
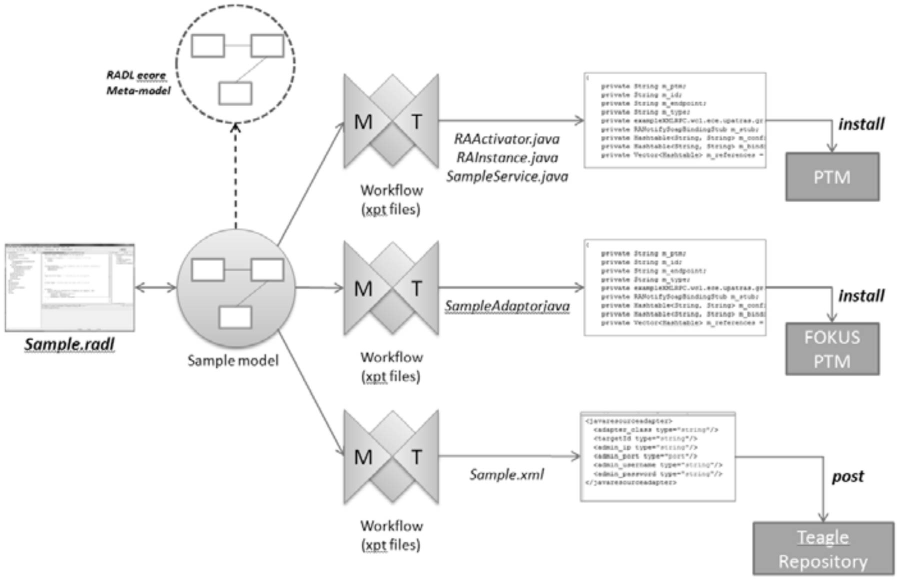


Fig. 5. The RADL meta-model as defined in Ecore

Currently, four APIs have been examined: SSH, HTTP, XML-RPC and a Java class. The `ConfigurationParams` are passed together with the `Binding Params` to the resource to be configured.



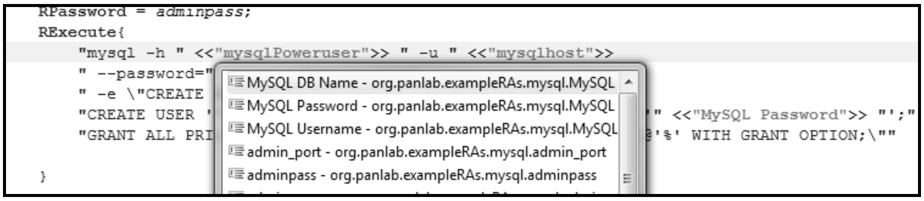
```
{
    private String m_ptm;
    private String m_id;
    private String m_endpoint;
    private String m_type;
    private exampleXMLRPC.wcl.ece.upatras.gr
    private RANotifySoapBindingStub m_stub;
    private Hashtable<String, String> m_conf
    private Hashtable<String, String> m_bind
    private Vector<Hashtable> m_references =
```

```
{
    private String m_ptm;
    private String m_id;
    private String m_endpoint;
    private String m_type;
    private exampleXMLRPC.wcl.ece.upatras.gr
    private RANotifySoapBindingStub m_stub;
    private Hashtable<String, String> m_conf
    private Hashtable<String, String> m_bind
    private Vector<Hashtable> m_references =
```

```
<javaresourceadapter>
  <adapter_class type="string"/>
  <targetId type="string"/>
  <admin_ip type="string"/>
  <admin_port type="port"/>
  <admin_username type="string"/>
  <admin_password type="string"/>
</javaresourceadapter>
```

Fig. 6. Practitioner's view and the artifacts of RADL

A practitioner's view and the artifacts are shown in Fig. 6. The RA practitioner defines the model of a RA in RADL by means of an editor implemented in Eclipse that supports the syntax of the language. While the practitioner describes the RA, the editor instantiates the RA model based on the RADL meta-model. For the definition of the concrete syntax of the language (and the Ecore model itself from the syntax of the language), the Textual Modeling Framework of Eclipse is used, specifically the XText framework [26]. Having defined the concrete syntax of RADL, the XText framework provides a rich editor with syntax-error detection and context assistance for the RADL practitioner, as depicted in Fig. 7.



```
RPassword = adminpass;
RExecute{
    "mysql -h " <<"mysqlPoweruser">> " -u " <<"mysqlhost">>
    " --password="
    " -e \"CREATE
    "CREATE USER '                              " <<"MySQL Password">> "';"
    "GRANT ALL PRI                              @'%' WITH GRANT OPTION;\""

}
```

MySQL DB Name - org.panlab.exampleRAs.mysql.MySQL
MySQL Password - org.panlab.exampleRAs.mysql.MySQL
MySQL Username - org.panlab.exampleRAs.mysql.MySQL
admin_port - org.panlab.exampleRAs.mysql.admin_port
adminpass - org.panlab.exampleRAs.mysql.adminpass

Fig. 7. The RADL editor with error detection and context assistance

Support tooling in Eclipse automatically generates the target code (currently Java) that implements the RA's model definition through model-to-text transformations. Such model-to-text transformations are written in xPand [27] template files. The practitioner initiates a corresponding workflow that triggers the generation of code by selecting the target Domain Manager (PTM) that should support the RA. There is also the possibility to post the description directly to the Teagle Repository (see section 2).

An example of the concrete syntax as defined in Xtext is as follows:

```
ruleSSH returns ruleSSH:
 {ruleSSH}'SSH' '{'
  'Remote Machine' '='
remoteMachine=[scriptParam|STRING] ';'
  'RPort' '=' RPort=[scriptParam|STRING] ';'
  'RUsername' '=' username=[scriptParam|STRING] ';'
  'RPassword' '=' password=[scriptParam|STRING] ';'
  'RExecute' '{' ( commands+=rulSSH_commands )*'}'
'}';
```

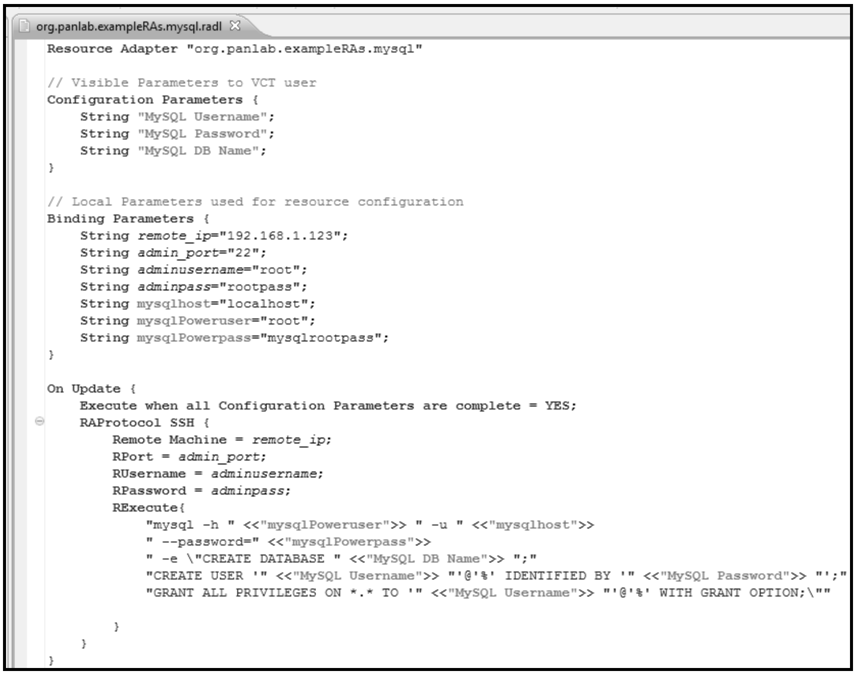
The above Xtext rule describes the way one can define an SSH wrapper command for a Resource Adapter. In quotation marks we define keywords of the language. More detailed use cases instantiating this SSH rule for a mysql Resource Adapter and the Java rule of the Amazon RA example are given in the next section.

4 Use Cases

In this section, two scenarios of offered resources are discussed. The first case presents the offering of a mySQL database. The second case demonstrates a generic HTTP POST resource for a target Domain Manager.

4.1 A MySQL Resource Adapter

Assuming that an organization wishes to offer a MySQL Database resource to the Panlab Federation, the organization must develop an RA and then publish some configuration parameters for the end-users: `MySQL Username`, `MySQL Password` and `MySQL DB Name`. The MySQL server is hosted by a Linux server, located at a machine with private IP 192.168.1.123.



```
Resource Adapter "org.panlab.exampleRAs.mysql"

// Visible Parameters to VCT user
Configuration Parameters {
    String "MySQL Username";
    String "MySQL Password";
    String "MySQL DB Name";
}

// Local Parameters used for resource configuration
Binding Parameters {
    String remote_ip="192.168.1.123";
    String admin_port="22";
    String adminusername="root";
    String adminpass="rootpass";
    String mysqlhost="localhost";
    String mysqlPoweruser="root";
    String mysqlPowerpass="mysqlrootpass";
}

On Update {
    Execute when all Configuration Parameters are complete = YES;
    RAProtocol SSH {
        Remote Machine = remote_ip;
        RPort = admin_port;
        RUsername = adminusername;
        RPassword = adminpass;
        RExecute{
            "mysql -h " <<"mysqlPoweruser">> " -u " <<"mysqlhost">>
            " --password=" <<"mysqlPowerpass">>
            " -e \"CREATE DATABASE " <<"MySQL DB Name">> ";"
            "CREATE USER '" <<"MySQL Username">> "'@'%' IDENTIFIED BY '" <<"MySQL Password">> "';"
            "GRANT ALL PRIVILEGES ON *.* TO '" <<"MySQL Username">> "'@'%' WITH GRANT OPTION;\""

        }
    }
}
```

Fig. 8. The RADL of the MySQL resource

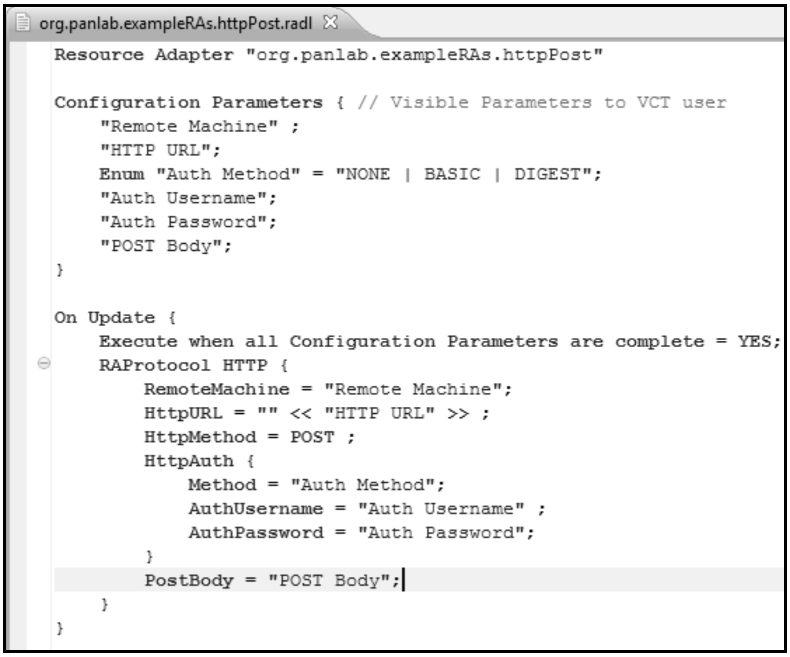
Fig. 8 displays the RADL syntax of this MySQL RA. The `Configuration Parameters` are the values exposed to the end user and appear on Teagle. The `Binding Parameters` are values that are configured by the RA developer: `remote_ip`, `admin_port`, `adminusername`, `adminpass` for connecting to the remote machine. The final parameters `mysqlhost`, `mysqlpoweruser` and `mysqlPowerpass` are for connecting to and configuring the MySQL resource.

The actions taken by the RA when it receives an UPDATE command (this is usually received by a Domain Manager on its generic interface towards the federation and is passed on to the RA inside a specific domain), is described in the `On Update` section. The RA will connect through SSH and will configure the MySQL resource as needed through some commands executed remotely.

The RADL workflow generates the necessary Java code for the target Domain Manager. For the Panlab PTM, almost 800 lines of Java code are automatically generated. For example the remote configuration commands are located at the method:

```
applyConf_SSH(){
String cmd = "";
cmd += "mysql -h ";
cmd += " " + m_configuration.get("mysqlPoweruser");
cmd += " -u ";
cmd += " " + m_configuration.get("mysqlhost");
...
```

For the Fraunhofer FOKUS Domain Manager (another implementation of a PTM) an equivalent Java code is generated (~350 lines of Java code). A similar `applyConf_SSH` method exists also in the Fraunhofer Domain Manager, although the configuration and binding parameters are handled differently.



org.panlab.exampleRAs.httpPost.radl

```
Resource Adapter "org.panlab.exampleRAs.httpPost"

Configuration Parameters { // Visible Parameters to VCT user
    "Remote Machine" ;
    "HTTP URL";
    Enum "Auth Method" = "NONE | BASIC | DIGEST";
    "Auth Username";
    "Auth Password";
    "POST Body";
}

On Update {
    Execute when all Configuration Parameters are complete = YES;
    RAProtocol HTTP {
        RemoteMachine = "Remote Machine";
        HttpURL = "" << "HTTP URL" >> ;
        HttpMethod = POST ;
        HttpAuth {
            Method = "Auth Method";
            AuthUsername = "Auth Username" ;
            AuthPassword = "Auth Password";
        }
        PostBody = "POST Body";
    }
}
```

Fig. 9. The RADL of the Generic HTTP POST Resource

4.2 A Generic HTTP POST Resource Adapter

In the next scenario, an organization wishes to offer a generic HTTP POST resource to an arbitrary URL. This means that the user simply needs to configure the remote machine and the HTTP URL where the POST is going to be executed. The user needs to optionally configure an authentication method (enumeration NONE, BASIC or DIGEST) with optional username and password. Finally, the user must define the POST body, for example a XML description. The RADL description for this resource is presented in Fig. 9. The code for the target Domain Manager can then be generated automatically.

4.3 Wrapping a Java Class for an Amazon Machine Instance Resource Adapter

As discussed earlier, a Resource Adapter wraps the API of a resource and exposes an interface to the underlying PTM and eventually to the user defining a VCT. With RADL we created a Resource Adapter for an Amazon's Machine Instance through the java EC2 API. This enables us to create Virtual Machines (VM) on the Amazon's cloud, configure them, and later on use them in federation scenarios. Although Amazon provides a plethora of settings for creating a VM, we wanted to provide a simple set of parameters to the end user. Also, we have developed a java class called EC2Wrapper, that has some methods for creating VMs on Amazon's cloud using the Java EC2 API. The following is the definition of the RA using the RADL syntax.

```
Resource Adapter "ami_ec2_ra"

Configuration Parameters { // Visible Parameters to VCT
user
 String AMI_Id = "ami-2cb05345" description = "An AMI
from Amazon list" ;
 String accessKey; //by amazon account
 String secretKey;//by amazon account
 String InstanceType = "m1.small" description = "AMI
type";
 String AvailabilityZone = "us-east-1a" description =
"AMI Region";
 String PublicDnsName description = "ReadOnly.
Available after creating VM";
 String loginUsername;
 String loginPassword;
 Integer maxNumberOfInstances= "1";//default is 1 AM
instance
}

On Update {
 ProcessOnAllConfigurationParametersComplete = YES;
 RAProtocol Java EC2Wrapper(accessKey, secretKey
){//Call the EC2Wrapper class
 JExecute createAMInstances(  AMI_Id , 1,
maxNumberOfInstances, loginUsername,
InstanceType , AvailabilityZone  )
 JAssign  AvailabilityZone = getPublicDnsName()
  }
}
```

The configuration parameters exposed to the user are presented in the section of **`Configuration Parameters`**. For example the `AMI_Id` can be set by the user to define which type of VM is required. To create VMs, the user is provided with the parameters `accessKey` and `secretKey`. Of particular interest is the `PublicDnsName` parameter, which at the end of the creation of the VM image will contain the assigned public DNS name from Amazon. This is used by other resources or by the end user to connect.

Another interesting part is the **On Update** event. For this RA, a Java class is wrapped, so we define the **RAProtocol Java** EC2Wrapper, which will wrap the EC2Wrapper class. This allows one to execute commands by passing user defined parameters, as shown in the line of **JExecute** createAMInstances, where parameters are available by the Configuration Parameters section. It is also possible to assign values to RA Configuration Parameters with the **JAssign** command. This rationale is shown by the last line, where the return value of the method getPublicDnsName() will be assigned to the AvailabilityZone parameter.

The result of the RADL environment in Eclipse, is to automatically create almost 1000 lines of java code describing the RA, which can be used immediately for installation in the PTM without user involvement. The following lines display how the **On Update** section of RADL was transformed automatically in java:

```
private void applyConf_JavaWrapper() {
 try {
ec2wrapper = new
EC2Wrapper(m_configuration.get("accessKey"),
       m_configuration.get("secretKey"));
  //JExecute

ec2wrapper.createAMInstances(m_configuration.get("AMI_Id"
), 1,

Integer.parseInt(m_configuration.get("maxNumberOfInstance
s")),
                m_configuration.get("loginUsername"),
                m_configuration.get("InstanceType"),
                m_configuration.get("AvailabilityZone"));
  //JAssignment
  m_configuration.put("AvailabilityZone",
ec2wrapper.getPublicDnsName());

 } catch (Exception e1) {
   // TODO Auto-generated catch block
   e1.printStackTrace();
 }
}
```

Fig. 10 displays an overview of how the Amazon RA works internally. After the creation of the RA in Java using the Eclipse RADL environment, the RA can be downloaded to the PTM of a testbed with other resources. Then it is made available to the Teagle repository in order to be used by the VCT tool. When a user creates a VCT, they can choose as many AMIs (Amazon Machine Image) as required and configure them accordingly.

When the provisioning starts, the AMI RA is instantiated. Internally, the EC2Wrapper class is instantiated where it starts negotiating with Amazon by utilizing the EC2 API through Amazon's web services. This is done by following a sequence of commands for creating the VM image, keypairs and reserving it.

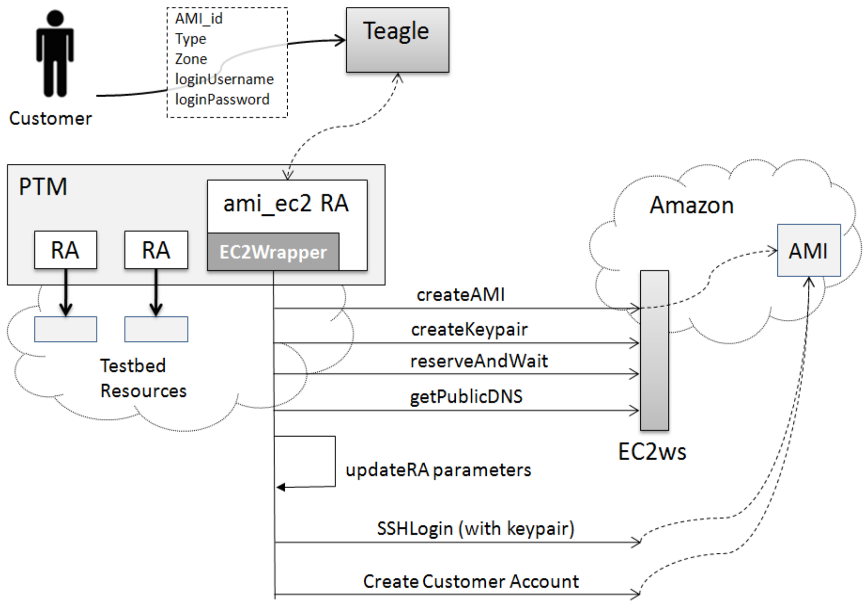


Fig. 10. How the Amazon RA works

When complete, there is an option to hand it over to the Panlab customer that owns the VCT, configured with the requested credentials, if the user didn't provide their Amazon credentials. This means that the testbed resource acts as a broker for Amazon, resulting in additional cost fees introduced by the testbed provider. More details regarding the Amazon RA can be found at [28].

5 Conclusion and Future Work

RADL is still under continuous development. However, first results are really encouraging as more people adopt it to develop RAs for the Panlab federation. As a starting point, it can be used for the RA practitioner by simply creating the code skeleton of an RA and continue developing in the target language.

More protocols, some workflow logic, and cleanup process on all generic CRUD (create, read, update, delete) commands that can be received by a Domain Manager on its federation interface, are under development. The syntax is also under continuous refactoring as practitioners work with the language. RADL is licensed under the Apache License, Version 2.0. More details, instructions, and downloads of RADL are available at http://trac.panlab.net/trac/wiki/RADL.

As the Panlab federation is currently transitioning from research project status to a production network with an increasing number of federation partners, the number of Domain Manager and Resource Adaptor implementations is also increasing. Thus, RADL will play a major role to ensure interoperability between different implementations and maximize the re-use of existing work, which has always been our motivation to federate ICT resources in the first place.

Acknowledgments. Parts of this work received funding by the European Commission's Sixth Framework Programme under grant agreement no.: 224119. We would like to thank the Panlab and PII consortia for the good collaboration as well as Prof. Dr. Paul Müller (Technische Universität Kaiserslautern/G-Lab) for the discussions on federation.

References

[1] Wahle, S., Magedanz, T., Gavras, A.: Conceptual Design and Use Cases for a FIRE Resource Federation Framework. In: Towards the Future Internet - Emerging Trends from European Research, pp. 51–62. IOS Press (2010)

[2] Wahle, S., et al.: Emerging testing trends and the panlab enabling infrastructure. IEEE Communications Magazine 49(3), 167–175 (2011)

[3] Wahle, S., Magedanz, T., Fox, S., Power, E.: Heterogeous resource description and management in generic resource federation frameworks. In: Proceedings of the 1st IFIP/IEEE Workshop on Managing Federations and Cooperative Management (May 2011) (to appear)

[4] National Science Foundation, GENI website, `http://www.geni.net`

[5] National Science Foundation, FIND website, `http://www.nets-find.net`

[6] European Commission, FIRE website, `http://cordis.europa.eu/fp7/ict/fire`

[7] Gavras, A., Karila, A., Fdida, S., May, M., Potts, M.: Future internet research and experimentation: the FIRE initiative. SIGCOMM Comput. Commun. Rev. 37(3), 89–92 (2007)

[8] AKARI project website, `http://akari-project.nict.go.jp/eng/index2.htm`

[9] Faber, T., Wroclawski, J., Lahey, K.: A DETER Federation Architecture. In: DETER Community Workshop on Cyber-Security and Test (2007)

[10] Peterson, L., Roscoe, T.: The Design Principles of PlanetLab. SIGOPS Oper. Syst. Rev. 40(1), 11–16 (2006)

[11] GENI Project Office, ProtoGENI Control Framework Overview, GENI-SE-CF-PGO-01.4 (2009)

[12] Chase, J., et al.: Beyond Virtual Data Centers: Toward an Open Resource Control Architecture. Selected Papers from the International Conference on the Virtual Computing Initiative. ACM Digital Library (2007)

[13] Ott, M., Seskar, I., Siraccusa, R., Singh, M.: ORBIT testbed software architecture: supporting experiments as a service. Testbeds and Research Infrastructures for the Development of Networks and Communities, 136– 145 (2005)

[14] OneLab project website, `http://www.onelab.eu/`

[15] Sezgedi, P., Figuerola, S., Campanella, M., Maglaris, V., Cervello-Pastor, C.: With Evolution for Revolution: Managing FEDERICA for Future Internet Research. IEEE Communications Magazine 47(7), 34–39 (2009)

[16] Gavras, A., et al.: Control of Resources in Pan-European Testbed Federation. In: Towards the Future Internet - A European Research Perspective, pp. 67–78. IOS Press (2009)

[17] Website of Panlab and PII European projects, supported by the European Commission in its both framework programmes FP6 (2001-2006) and FP7 (2007-2013), `http://www.panlab.net`

[18] Asia-Pacific Advanced Network initiative website, `http://www.apan.net`
[19] Asia Future Internet Forum website, `http://www.asiafi.net`
[20] Chen, M., Moon, S., Nakao, A.: Goals and Blueprint for PlanetLab CJK. Presentation at Conference for Future Internet 2008 PlanetLab BoF, Seoul, Korea, June 19 (2008)
[21] Magedanz, T., Wahle, S.: Control Framework Design for Future Internet Testbeds. e & i Elektrotechnik und Informationstechnik 126(07/08), 274–279 (2009)
[22] Strassner, J.: Policy-Based Network Management. Morgan Kaufmann Publishers (2003) ISBN: 1-55860-859-1
[23] Wahle, S., Harjoc, B., Campowsky, K., Magedanz, T., Gavras, A.: Pan-European testbed and experimental facility federation - architecture refinement and implementation. International Journal of Communication Networks and Distributed Systems (IJCNDS), Special Issue on Recent Advances in Testbed Driven Networking Research 5(1/2), 67–87 (2010)
[24] OMG website, Catalog of OMG Modeling and Metadata Specifications, `http://www.omg.org/technology/documents/modeling_spec_catalog.htm`
[25] Eclipse Foundation website, `http://www.eclipse.org/modeling/emf/?project=emf#emf`
[26] TMF, XText framework website, `http://www.eclipse.org/XText`
[27] XPand website, M2T, XPand statically typed tempaler language, `http://www.eclipse.org/modeling/m2t/?project=xpand`
[28] ami_ec2: A Resource Adapter for creating Amazon Machine Instances in the Amazon EC2, `http://trac.panlab.net/trac/wiki/AMI_EC2`
[29] Power, E., Boudjemil, Z., Fox, S.: Architecture & Implementation of a Testbeds Repository. In: International Conference on Telecommunications and Multimedia (2010)