PROLOGUE

CompatibleOne

The Accords Platform
Version 1.3

Iain James Marshall

11/30/2012

This document describes the different deployment models of the Advanced Capabilities of the CompatibleOne Resources Description System known as the Accords Platform.

Table of Contents

Introduction

This document describes the different deployment models of the Advanced Capabilities of the CompatibleOne Resources Description System known as the Accords Platform. The description of the Accords Platform is in terms of its architecture, its access protocol, its monitoring systems and the various deployment models that may be employed in order to achieve autonomous operation. The production, installation and configuration of the platform will be described in detail in order to provide a solid understanding of the basic principles involved for the small, large and scalable models.

The paths used for the installation of the small model Accords Platform correspond to the extraction of the sources from the open source repository. These may be different if the package is installed from a Debian package or a Red Hat package.

Small Accords Platform

The basis of the autonomous operation of the Accords Platform reposes on the principle of a small Accords Platform instance responsible for the deployment and management of the large operational, and eventually fully scalable, version of the platform. The small platform is intended to be monitored using third party open source monitoring software such as NAGIOS for which configuration is already possible.

Installation

The extraction of the sources from the OW2 Github Source Repository followed by their compilation allows the construction of the binary executable programs and shared libraries of the base kernel version of the Accords Platform.

The paths used for the installation of the small model Accords Platform correspond to the extraction of the sources from the open source repository. These may be different if the package is installed from a Debian package or a Red Hat package. Please consult the documentation corresponding to the appropriate package for more information in this respect.

The following sequence of LINUX shell commands shows the way in which this may be accomplished:

```
$ cd /home
$ git clone ssh://git@gitorious.ow2.org/ow2-compatibleone/accords-platform.git
$ ./autogen.sh
$ ./configure
$ make clean
$ make
$ make install
```

The resulting components of this “Small Accords Platform”, as it is known, will be generated and deposited into the standard LINUX directories *“/usr/local/bin*” and *“/usr/local/lib*”.

Before the platform components can be activated it is necessary to prepare an operating environment in the form of a working directory. It is strongly recommended that this be positioned outside of the source tree directory that was extracted from the source repository. An example of LINUX shell commands for the configuration of this environment can be seen below:

```
$ mkdir /home/small-accords
$ mkdir /home/small-accords/rest
$ mkdir /home/small-accords/service
$ cd /home/small-accords
$ cp /home/accords-platform/scripts/accords-config /home/small-accords
$ cp /home/accords-platform/scripts/cords_user.xml /home/small-accords
$ cp /home/accords-platform/manifests/cords_price.xml /home/small-accords
$ cp /home/accords-platform/scripts/style.css /home/small-accords
$ cp /home/accords-platform/scripts/openssl.cnf /home/small-accords
$ cp /home/accords-platform/coips.xml /home/small-accords
$ cp /home/accords-platform/accounts.xml /home/small-accords
$ cp /home/accords-platform/metrics.xml /home/small-accords
$ cp /home/accords-platform/Certificates.crt /home/small-accords
```

As can be seen above, several files are to be transferred from the source package directories and may require modification as described below:

- The file “*cords_user.xml*” must contain an entry for each of the category manager services that are to be activated for the platform. The file present in the scripts source directory contains the definitions of all the users required for normal platform operation.

- The file "*cords_price.xml*" provides the basic pricing structure and the version copied here from the scripts source directory serves as an example.
- The file "*metrics.xml*" contains the description of a collection of metric values that can be used for monitoring and service level agreement control.
- The file "*accounts.xml*" provides the default collection of user accounts required for platform operation.
- The file "*coips.xml*" provides the different descriptions of infrastructure that will be used by the image production service when building images on the different platform types.
- The files "*openssl.cnf*" and "*Certificates.crt*" are only needed only if security is to be activated.

In addition to the above files, subscription account information required by default for provisioning will need to be provided in the files "*os_config.xml*", "*on_config.xml*", "*az_config.xml*". This information is to be specified using a text editor by the operator of the small accords platform.

Configuration

The configuration of the ensemble is generated and activated during the execution of the primary installation script, "*accords-config*". Prior to launching this script it may be necessary to adapt it to your particular platform and system usage requirements. This installation script is, however, generated during the platform production phase and it is recommended to adapt the original source file "accords.in" to reflect your operating conditions and requirements and then update the installation by issuing the standard "`make; make install`" commands. The following environment variables are to be positioned as described below:

- **TLS**

  This environment variable should be set to indicate the path name of the security directory to be used for the production and storage of the certificates, private keys and security configuration files of the component servers of the small accords platform. When this variable defines an empty string then security is dis-activated and the small accords platform will operate in insecure HTTP mode. This is not recommended but may be necessary from time to time for debug reasons.

- **RESTHOST**

  This environment variable is to be set to the public IP address or public domain name of the machine on which the small accords platform will operate. This is essential when security has been activated in order to allow the COSACS interfaces of deployed virtual machines to authenticate and procure authorization prior to commencing their activity.

- **NETHSM**

  This environment variable may be set to 1 to indicate that the certificates and private keys are to be stored in an external hardware security device.

- **NAGIOS**

  This environment variable should be set to specify the full path and file name of the file to which the NAGIOS configuration information is to be written as the indication that the

NAGIOS monitoring system is to be used for the monitoring of the small accords platform. The appropriate configuration information will be generated, for which the NAGIOS system will be restarted, when this variable has been set. The current versions of NAGIOS are not TLSv1 compliant. This may require the modification of the security options of the Accords Platform configuration script accordingly from the standard value of 13 to the value of 141 allowing insecure secure communication using SSLv23. This is not recommended in production environments.

With the required values for these environment variables having been correctly specified then the configuration script may be launched to produce the "*accords.xml*" configuration file containing the configuration information for the complete collection of category manager OCCI servers comprising the platform.

```
$ ./accords-config
```

The small platform may then be started:

```
$ co-start
```

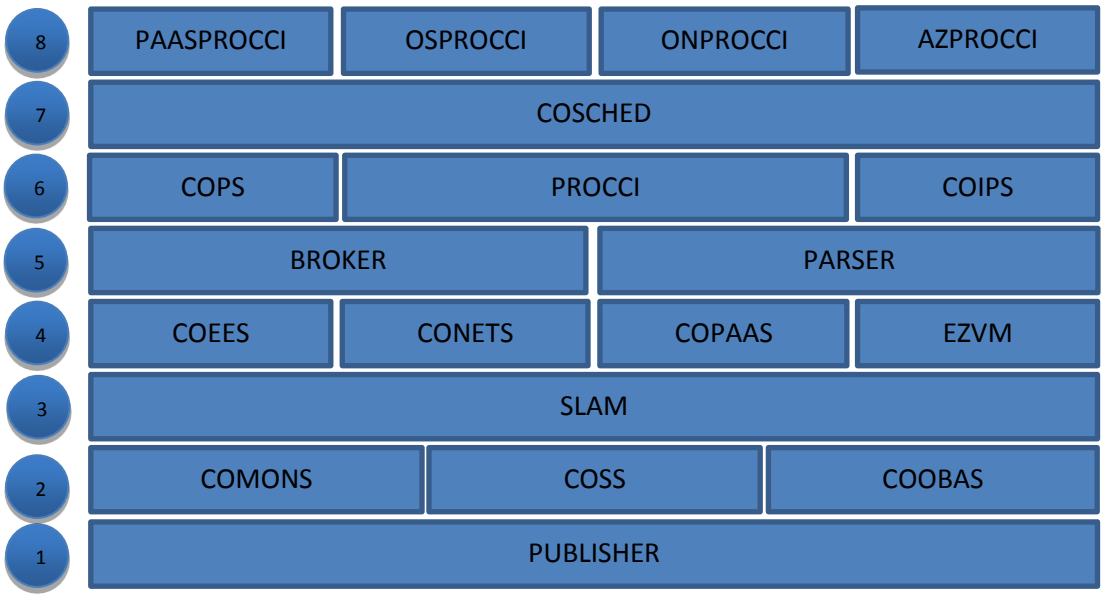
Each of the basic components will be started and then, for each of the provisioning interfaces, the provider service level agreement, account and quota, will be generated, prior to activation of the corresponding provider interface.

Once all components of the platform have been started then the operational conformity and availability of the required category manager components may be verified using:

```
$ co-check
```

Architecture

The completion of the above described procedure will result in the collection of OCCI Category Management REST Server components being activated and brought ONLINE. This collection of components is capable of performing provisioning operations resulting from the parsing of resource description manifests and the brokering of provisioning plans for the deployment of instances. The following diagram depicts the relationship between the various platform components.



1: Publication Layer

The platform architecture is built on top of the services provided by the publisher. This component provides a low level, OCCI category, service publication service through which all other components of the platform make known their offers of service in terms of the OCCI categories that they manage. Likewise, the publisher will be consulted, by all components of the platform, for the discovery and resolution of the access addresses of the category managers with which they need to communicate during the course of their own particular operations.

2: Access Protocol Layer

The first layer, on top of the publication layer, comprises the security, monitoring and financial services that provide the basis for authentication, authorization, audit, event logging and transaction management and constitute the mandatory elements of the Accords Platform Access Protocol.

3: Service Level Management Layer

The service level agreement layer is built on top of the access protocol support layer and provides the descriptive and operational elements required for the support and management of customer service level agreements and provider service level agreements.

4: Service Layer

On top of the service level agreement layer we can see the service layer comprising components for the management and support of diverse offers of service such as Platform as a Service, Network as a Service, Energy Efficiency as a Service and Virtual Machine Image Management as a Service. The

addition of new service management types ranging from Software as a Service through Business Processes as a Service would be performed by the extension of this layer.

5: Functional Layer

The functional layer of the platform comprising the parsing and brokering services follows next. These two services are fundamental to the platform's operation. The parser provides a completely extensible resource description, processing and management system for the introduction of information and operational data to the system. Information and data are to be described in XML documents referred to as manifests. Manifests are to be defined by their XSD schema and are processed by the parsing services and tools. Information and data resulting from this operation will be stored as OCCI category instances within the Accords Platform. The broker provides a fully extensible resource deployment mechanism whereby descriptions resulting from the parsing phase, known as provisioning plans, are processed to produce a service instance management graph. This operation is orchestrated by the broker working in conjunction with the preceding service layer and the subsequent contract management layer.

6: Contract Management Layer

This layer comprises the placement services, the image production services and the generic contract management services and is responsible for the management and coordination of the various stages in the contract management life cycle comprising the following four states:

1. **Creation**

   Here the generic contract is created through the appropriate provider specific interface using the technical and commercial description of the resources that are required. The resulting contract will be appended to the corresponding service control graph.

2. **Deployment**

   Here the reserved resources are provisioned, deployed and effectively consumed and eventually temporal periodic pricing and costing operations will commence the delivery of transactions for the corresponding account.

3. **Release**

   Provisioned and deployed resources will be released and pricing and costing will be calculated and completed.

4. **Deletion**

   Reserved provider resources will be restituted to the corresponding provider quota pool and contract management structures will be removed.

7: Scheduling Layer

The scheduling layer is responsible for the coordination of deployment requests to the provisioning layer issued by the contract management layer.

8: Provisioning Layer

The provisioning layer comprises the collection of provider specific interfaces required for the deployment of resources. The addition of new interface components to cater for emerging provisioning technology types would occur in this layer of the overall architecture.

Accords Platform Access Protocol

The term “Accords Platform Access Protocol” refers to the operational procedure that is to be respected by all components wishing to participate in platform operation. This comprises the following stages:

Transport

All components are required to communicate using Transport Layer Security Version 1.0 only and must present a valid PKI Certificate suitable signed by a known Certification Authority.

Authentication

All components are required to present valid user credentials, comprising login name and password, for authentication by the security services. These credentials allow determination not only of the role of the endpoint from amongst agent, user, administrator, provider or operator but also of the access permissions in terms of discrete record management operations. Further work will be required in this area for the adjunction of the role based policy engine initially intended to be provided by the ORBAC system.

Authorization

All components are required to procure an authorization token, through the security services, using their authenticated credentials. This authorization token is to be presented in the request and response headers of all messages exchanged during the component’s participation in platform operation. Authorization credentials are expected, though not required, to be validated by all recipients.

Accountability

All components are required to log their participation using the means identified and made available by the configuration of the platform. In small model, this will be the system log file. In large model, the centralized event service will be used. It is recommended that, in large model, log information is also to be written to the local log file. Information logged, in either way, must include a record for each request received, for each response returned, each request sent and for each response received. In this way the complete sequence of inter-operation is captured for further investigated at any moment in time.

Publication

With the three preceding layers of AAA protocol in place the component may proceed to the final stage concerning the publication of service. Accords Platform components are OCCI Category Servers operating in a RESTful manner over HTTP. Categories that are offered for use by other components, said to be public categories, are to be published through the platform publication service known as the publisher. Certain categories, representing provisioning concepts, are to be published through the provider interface. Finally categories representing monitoring endpoints are to be published through the monitoring consumer interface. Each of these publication records may be accompanied by financial information allowing for costing and pricing to be performed before, during and after engagement of the service interface by eventual client endpoints.

Operation

Once the publication phase has completed the component may commence its participation in the platform activity by launching the corresponding OCCI/REST/HTTP server for the reception and processing of category request messages. Each individual server component is an autonomous multi-threaded device incorporating thread limit control. This allows for the management of scale by the component when operating in a suitably enabled environment.

Shutdown

All components are responsible for performing a clean and organized shutdown. This entails the cancellation of all publications including through provider and consumer interfaces. The authorization credential is then to be de-validated and destroyed.

Large Accords Platform

This version of the Accords Platform, the large version, involving the deployment of each individual component of the small model in its own virtual machine, is to be deployed and managed by an instance of the Small Accords Platform. The large model Accords Platform comprises a collection of manifests, describing the various components of the platform, to be parsed and brokered through the Small Accords Platform. Each manifest will perform installation of the corresponding Accords Platform component, through the COSACS interface, after the service contract deployment of resources in response to the invocation of the "*start*" action. This installation procedure requires the retrieval of the source component package from a centralized depot. These packages will be produced and uploaded to the depot during the creation of the manifests describing the large model Accords Platform. The following sequence of LINUX commands shows the way in which this collection of manifests, and their subordinate packages, may be produced for a particular operating system and provisioning platform. This will be performed for deployment and use through a particular user account:

```
$ cd /home/accords-platform/manifests
$ bash make-all <account> <security>
```

The environment variable or parameter named "**<account>**" above must be set to provide the name of the Small Accords Platform User Account through which eventual deployment is to be performed.

The environment variable or parameter named "**<security>**" above must be set to provide either the value "security" or left blank if secure operation is not required.

During operation of the "*make-all*" script, the operator will be required to validate in order to confirm the desire to proceed to the selection of the base operating system type. It will be necessary to specify one of the proposed values of "Debian, ubuntu, ubuntu64 etc" by typing the value in at the prompt and validating. Selection of one of the provisioning platform types from the proposed list of values "openstack, opennebula, windowsazure etc" must then be performed by typing in the value at the prompt and validating. The collection of manifests will then be produced and packaged into the file "accords-manifests.tgz" that will be transferred, along will all ancillary support packages, up to the central Accords Platform depot. The sources extracted from the open source repository are for use by authorized persons only since it is required that an authorization password be specified prior to allowing the delivery of these packages to the depot. For a personalized use of the package delivery system simply modify the URL address of the targeted depot, in the corresponding script, to specify the address of the required private depot.

The collection of manifests, describing the large platform, having been suitably prepared, using the large model manifest generation tool provided in the source package, are to be parsed into the small platform. The appropriate service level agreements will then be generated using the SLA generation tools, build-manifest or build-guarantee, depending on whether guarantee clauses are required. The resulting agreement, one for the platform and one for the provisioning, will then be parsed and their provisioning plans used, in sequence, for the deployment of the resources via the small accords platform broker. The resulting multi virtual machine configuration may then be used for the parsing of manifests and the brokering of instances.

The collection of manifests describing the large accords platform will now be examined in detail. They all follow the same basic lines and each represents a single OCCI Category Management Component to be deployed in one single virtual machine.

PUBLISHER

As for the small accords platform, the publisher provides the base on which the rest of the platform reposes. It is packed alone in its own manifest “accords-publisher.xml”, the contents of which can be seen below.

```
<manifest name="accords:publisher" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node   name="publisher" access="public" scope="normal" type="simple"
                provider="openstack">
                <infrastructure name="accords:component">
                <compute name="accords:component"
                        architecture="x86_64" cores="1" memory="1G" speed="1G"/>
                <storage name="accords:component" size="10G"/>
                <network name="PUB_Compatible" label="ethernet" vlan="100M">
                        <port name="ssh" protocol="tcp" from="22" to="22"/>
                        <port name="http" protocol="tcp" from="80" to="80"/>
                        <port name="cosacs" protocol="tcp" from="8286" to="8286"/>
                        <port name="publisher" protocol="tcp" from="8086" to="8086"
                        range="0.0.0.0/0"/>
                        <port name="platform" protocol="tcp" from="8087" to="8087"
                        range="0.0.0.0/0"/>
                </network>
                </infrastructure>
                <image name="debian:publisher">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="accords:publisher">
        <action expression="publisher.system('wget http://www.compatibleone.fr/accords-
        platform/debian/run-publisher');"/>
        <action expression="publisher.fork('bash run-publisher');"/>
</configuration>
<release name="accords:publisher">
        <action expression="publisher.kill('publisher');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest sets the scene for all subsequence manifests by defining the standard accords platform component infrastructure component and image components. The configuration section retrieves the publisher package and runs the installation script. The release section requests orderly termination of the publisher with release of all reserved resources.

COSS

The CompatibleOne Security Service component, COSS, is the next for deployment and establishes the conditions for subsequent modules compliant with the security oriented features of the above described Accords Platform Access Protocol. This manifest is packaged in the file “accords-coss.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:coss" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="coss" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:coss">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:coss">
        <action expression="coss.configure(accords:publisher.publisher.hostname);"/>
```

```
        <action expression="coss.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-coss');"/>
        <action expression="coss.fork('bash run-coss');"/>
</configuration>
<release name="debian:coss">
        <action expression="coss.kill('coss');"/>
        <action expression="coss.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

Attention is drawn here to the following points:

- The manifest describes two nodes.
- The first node is described by the manifest of the accords publisher and is of “public common” deployment nature. As such it will exist as a singleton instance to be shared by all instances of manifests which reference this same component node.
- This manifest introduces the idea that subsequent manifests will make use of the infrastructure and image definitions performed in the preceding manifest of the publisher component.
- The configuration section shows the way in which the address of the publisher is communication through the configuration action expression for use by the security service component. In this way the security service is able to connect to the publisher for the publication of the security protocol interface categories.

COMONS

The CompatibleOne Monitoring Service component, COMONS, is the next for deployment and completes the conditions for subsequent modules compliant with the accountability features of the above described Accords Platform Access Protocol. This manifest is packaged in the file “accords-comons.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:comons" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="comons" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:comons">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:comons">
        <action expression="comons.configure(accords:publisher.publisher.hostname);"/>
        <action expression="comons.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-comons');"/>
        <action expression="comons.fork('bash run-comons');"/>
</configuration>
<release name="debian:comons">
        <action expression="comons.kill('comons');"/>
        <action expression="comons.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the COMONS server software package.

COOBAS

The CompatibleOne Ordering, Billing and Accounting Service component, COOBAS, is the next for deployment and is packaged in the manifest file “accords-coobas.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:coobas" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="coobas" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:coobas">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:coobas">
        <action expression="coobas.configure(accords:publisher.publisher.hostname);"/>
        <action expression="coobas.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-coobas');"/>
        <action expression="coobas.fork('bash run-coobas');"/>
</configuration>
<release name="debian:coobas">
        <action expression="coobas.kill('coobas');"/>
        <action expression="coobas.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the COOBAS server software package.

SLAM

The CompatibleOne Service Level Agreement Management service component, SLAM, is the next for deployment and is packaged in the manifest file “accords-slam.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:slam" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="slam" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:slam">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:slam">
        <action expression="slam.configure(accords:publisher.publisher.hostname);"/>
        <action expression="slam.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-slam');"/>
        <action expression="slam.fork('bash run-slam');"/>
</configuration>
<release name="debian:slam">
        <action expression="slam.kill('slam');"/>
        <action expression="slam.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the SLAM server software package.

CONETS

The CompatibleOne Network Service component, CONETS, is the next for deployment and is packaged in the manifest file “accords-conets.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:conets" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="conets" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:conets">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:conets">
        <action expression="conets.configure(accords:publisher.publisher.hostname);"/>
        <action expression="conets.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-conets');"/>
        <action expression="conets.fork('bash run-conets');"/>
</configuration>
<release name="debian:conets">
        <action expression="conets.kill('conets');"/>
        <action expression="conets.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the CONETS server software package.

COEES

The CompatibleOne Energy Efficiency Service component, COEES, is the next for deployment and is packaged in the manifest file “accords-coees.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:coees" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="coees" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:coees">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:coees">
        <action expression="coees.configure(accords:publisher.publisher.hostname);"/>
        <action expression="coees.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-coees');"/>
        <action expression="coees.fork('bash run-coees');"/>
</configuration>
<release name="debian:coees">
        <action expression="coees.kill('coees');"/>
        <action expression="coees.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the COEES server software package.

COPAAS

The CompatibleOne Platform as a Service component, COPAAS, is the next for deployment and is packaged in the manifest file “accords-copaas.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:copaas" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
```

```
        <node name="copaas" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:copaas">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:copaas">
        <action expression="copaas.configure(accords:publisher.publisher.hostname);"/>
        <action expression="copaas.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-copaas');"/>
        <action expression="copaas.fork('bash run-copaas');"/>
</configuration>
<release name="debian:copaas">
        <action expression="copaas.kill('copaas');"/>
        <action expression="copaas.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the COPAAS server software package.

EZVM

The CompatibleOne Virtual Machine service component, EZVM, is the next for deployment and is packaged in the manifest file “accords-ezvm.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:ezvm" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="ezvm" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:ezvm">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:ezvm">
        <action expression="ezvm.configure(accords:publisher.publisher.hostname);"/>
        <action expression="ezvm.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-ezvm');"/>
        <action expression="ezvm.fork('bash run-ezvm');"/>
</configuration>
<release name="debian:ezvm">
        <action expression="ezvm.kill('ezvm');"/>
        <action expression="ezvm.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the EZVM server software package.

COIPS

The CompatibleOne Image Production Service component, COIPS, is the next for deployment and is packaged in the manifest file “accords-coips.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:coips" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="coips" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:coips">
                        <system name="debian_with_cosacs"/>
```

```
                </image>
        </node>
<configuration name="debian:coips">
        <action expression="coips.configure(accords:publisher.publisher.hostname);"/>
        <action expression="coips.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-coips');"/>
        <action expression="coips.fork('bash run-coips');"/>
</configuration>
<release name="debian:coips">
        <action expression="coips.kill('coips');"/>
        <action expression="coips.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the COIPS server software package.

COPS

The CompatibleOne Placement Service component, COPS, is the next for deployment and is packaged in the manifest file “accords-cops.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:cops" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="cops" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:cops">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:cops">
        <action expression="cops.configure(accords:publisher.publisher.hostname);"/>
        <action expression="cops.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-cops');"/>
        <action expression="cops.fork('bash run-cops');"/>
</configuration>
<release name="debian:cops">
        <action expression="cops.kill('cops');"/>
        <action expression="cops.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the COPS server software package.

PROCCI

The CompatibleOne generic contract provisioning service component, PROCCI, is the next for deployment and is packaged in the manifest file “accords-procci.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:procci" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="procci" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:procci">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:procci">
        <action expression="procci.configure(accords:publisher.publisher.hostname);"/>
```

```
        <action expression="procci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-procci');"/>
        <action expression="procci.fork('bash run-procci');"/>
</configuration>
<release name="debian:procci">
        <action expression="procci.kill('procci');"/>
        <action expression="procci.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the PROCCI server software package.

PARSER

The CompatibleOne parsing service component, PARSER, is the next for deployment and is packaged in the manifest file “accords-parser.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:parser" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="parser" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:parser">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:parser">
        <action expression="parser.configure(accords:publisher.publisher.hostname);"/>
        <action expression="parser.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-parser');"/>
        <action expression="parser.fork('bash run-parser');"/>
</configuration>
<release name="debian:parser">
        <action expression="parser.kill('parser');"/>
        <action expression="parser.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the PARSER server software package.

BROKER

The CompatibleOne brokering service component, BROKER, is the next for deployment and is packaged in the manifest file “accords-broker.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:broker" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="broker" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:broker">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:broker">
        <action expression="broker.configure(accords:publisher.publisher.hostname);"/>
        <action expression="broker.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-broker');"/>
        <action expression="broker.fork('bash run-broker');"/>
</configuration>
<release name="debian:broker">
```

```
        <action expression="broker.kill('broker');"/>
        <action expression="broker.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the BROKER server software package.

COSCHED

The CompatibleOne Scheduling service component, COSCHED, is the next for deployment and is packaged in the manifest file “accords-cosched.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:cosched" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="cosched" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:cosched">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
<configuration name="debian:cosched">
        <action expression="cosched.configure(accords:publisher.publisher.hostname);"/>
        <action expression="cosched.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-cosched');"/>
        <action expression="cosched.fork('bash run-cosched');"/>
</configuration>
<release name="debian:cosched">
        <action expression="cosched.kill('cosched');"/>
        <action expression="cosched.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

This manifest is similar in structure to the preceding manifest for the security service with retrieval and launch of the COSCHED server software package.

PLATFORM

The complete collection of manifests defined so far is instanced as nodes within the following manifest describing the base services of the platform. This is packaged in the XML file “accords-platform.xml” and can be seen below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:platform" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
<node name="accords:coss" scope=normal access=public type="accords:coss"/>
<node name="accords:comons" scope=normal access=public type="accords:comons"/>
<node name="accords:coobas" scope=normal access=public type="accords:coobas"/>
<node name="accords:conets" scope=normal access=public type="accords:conets"/>
<node name="accords:cosched" scope=normal access=public type="accords:cosched"/>
<node name="accords:cops" scope=normal access=public type="accords:cops"/>
<node name="accords:coees" scope=normal access=public type="accords:coees"/>
<node name="accords:copaas" scope=normal access=public type="accords:copaas"/>
<node name="accords:slam" scope=normal access=public type="accords:slam"/>
<node name="accords:ezvm" scope=normal access=public type="accords:ezvm"/>
<node name="accords:coips" scope=normal access=public type="accords:coips"/>
<node name="accords:broker" scope=normal access=public type="accords:broker"/>
<node name="accords:parser" scope=normal access=public type="accords:parser"/>
<node name="accords:procci" scope=normal access=public type="accords:procci"/>
<configuration name="accords:platform"/>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

All nodes are declared public allowing eventual referencing and use from subsequent manifest derivations.

Deployment of the Large Accords Platform

The deployment of the large accords platform as described by the preceding collection of manifests may be performed using the collection of LINUX shell commands that can be seen below:

```
$ co-parser accords-publisher
$ co-parser accords-coss
$ co-parser accords-comons
$ co-parser accords-coobas
$ co-parser accords-slam
$ co-parser accords-coees
$ co-parser accords-conets
$ co-parser accords-copaas
$ co-parser accords-ezvm
$ co-parser accords-parser
$ co-parser accords-broker
$ co-parser accords-coips
$ co-parser accords-procci
$ co-parser accords-cops
$ co-parser accords-cosched
$ co-parser accords-platform
$ co-parser sla-accords-platform
$ co-broker sla-accords-platform
```

Upon successful completion an individual virtual machine will have been provisioned and deployed for each of the components and interconnected to operate around the shared common publisher instance. Please note that deployment will be for the account specified during the creation of the collection of manifests and associated service level agreement.

OSPROCCI

The first of the provisioning interface manifests is the OSPROCCI encapsulating the OpenStack NOVA interface for provisioning on OpenStack platforms. Although the PROCCI was initially designed for use with the CACTUS and DIABLO releases extensive modifications were necessary to ensure continued operation with both the ESSEX and FOLSOM releases. This manifest is packaged in the XML file “accords-osprocci.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:osprocci" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
<node name="accords:publisher" access="public" scope="common" type="accords:publisher"/>
<node name="osprocci" access="public" scope="normal" type="simple" provider="openstack">
<infrastructure name="accords:component"/>
<image name="debian:osprocci">
<system name="debian_with_cosacs"/>
</image>
</node>
<configuration name="debian:osprocci">
<action expression="osprocci.configure(accords:publisher.publisher.hostname);"/>
<action expression="osprocci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-tools');"/>
<action expression="osprocci.system('bash run-tools');"/>
<action expression="osprocci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-osprocci');"/>
<action expression="osprocci.fork('bash run-osprocci');"/>
</configuration>
<release name="debian:osprocci">
<action expression="osprocci.kill('osprocci');"/>
<action expression="osprocci.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

Here the standard retrieval of the server package is accompanied by the retrieval and installation of the parsing and brokering tools. This is necessary in the case of the provisioning manifests in order that the parsing of the account, quota and SLA manifest documents may be performed prior to the launch of the provisioning interface.

ONPROCCI

The second of the provisioning interface manifests encapsulates an OCCI Version 0.9 interface for access to the Open Nebula provisioning system. This is packaged in the XML file “accords-onprocci.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:onprocci" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
<node name="accords:publisher" access="public" scope="common" type="accords:publisher"/>
<node name="onprocci" access="public" scope="normal" type="simple" provider="openstack">
<infrastructure name="accords:component"/>
<image name="debian:onprocci">
<system name="debian_with_cosacs"/>
</image>
</node>
<configuration name="debian:onprocci">
<action expression="onprocci.configure(accords:publisher.publisher.hostname);"/>
<action expression="onprocci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-tools');"/>
<action expression="onprocci.system('bash run-tools');"/>
<action expression="onprocci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-onprocci');"/>
<action expression="onprocci.fork('bash run-onprocci');"/>
</configuration>
<release name="debian:onprocci">
<action expression="onprocci.kill('onprocci');"/>
<action expression="onprocci.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

As for the OpenStack PROCCI the retrieval of the server package is accompanied by the retrieval and installation of the parsing and brokering tools.

PAPROCCI

The third of the provisioning interface manifests encapsulates a JAVA interface for access to the PROACTIVE PACA GRID provisioning system. This is packaged in the XML file “accords-paprocci.xml”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:paprocci" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
<node name="accords:publisher" access="public" scope="common" type="accords:publisher"/>
<node name="paprocci" access="public" scope="normal" type="simple" provider="openstack">
<infrastructure name="accords:component"/>
<image name="debian:paprocci">
<system name="debian_with_cosacs"/>
</image>
</node>
<configuration name="debian:paprocci">
<action expression="paprocci.configure(accords:publisher.publisher.hostname);"/>
<action expression="paprocci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-tools');"/>
<action expression="paprocci.system('bash run-tools');"/>
<action expression="paprocci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-paprocci');"/>
<action expression="paprocci.fork('bash run-paprocci');"/>
</configuration>
<release name="debian:paprocci">
<action expression="paprocci.kill('paprocci');"/>
<action expression="paprocci.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
```

```
</manifest>
```

As for the OpenStack PROCCI the retrieval of the server package is accompanied by the retrieval and installation of the parsing and brokering tools.

PAASPROCCI

The fourth and final provisioning interface manifest encapsulates the generic PaaS interface for access to the compatible Platform as a Service provisioning systems. This is packaged in the XML file "accords-paasprocci.xml".

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:paasprocci" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
<node name="accords:publisher" access="public" scope="common" type="accords:publisher"/>
<node name="paasprocci" access="public" scope="normal" type="simple" provider="openstack">
<infrastructure name="accords:component"/>
<image name="debian:paasprocci">
<system name="debian_with_cosacs"/>
</image>
</node>
<configuration name="debian:paasprocci">
<action expression="paasprocci.configure(accords:publisher.publisher.hostname);"/>
<action expression="paasprocci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-tools');"/>
<action expression="paasprocci.system('bash run-tools');"/>
<action expression="paasprocci.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-paasprocci');"/>
<action expression="paasprocci.fork('bash run-paasprocci');"/>
</configuration>
<release name="debian:paasprocci">
<action expression="paasprocci.kill('paasprocci');"/>
<action expression="paasprocci.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

As for the OpenStack PROCCI the retrieval of the server package is accompanied by the retrieval and installation of the parsing and brokering tools.

PROVISIONING

The four provisioning interface manifests described above are deployed through the following manifests packaged in the XML file "accords-provisioning.xml".

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:provisioning"
xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
<node name="accords:osprocci" scope=normal access=public type="accords:osprocci"/>
<node name="accords:onprocci" scope=normal access=public type="accords:onprocci"/>
<node name="accords:paprocci" scope=normal access=public type="accords:paprocci"/>
<node name="accords:paasprocci" scope=normal access=public type="accords:paasprocci"/>
<configuration name="accords:provisioning"/>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

Again, as for the accords platform manifests, these nodes are declared public allowing eventual use by subsequent higher level derivations.

Deployment of the Large Model Accords Provisioning

The deployment of the large model accords platform provisioning as described by the preceding collection of manifests may be performed using the collection of LINUX shell commands that can be seen below:

```
$ co-parser accords-osprocci
$ co-parser accords-onprocci
```

```
$ co-parser accords-paprocci
$ co-parser accords-paasprocci
$ co-parser accords-provisioning
$ co-broker accords-provisioning
$ co-parser sla-accords-provisioning
$ co-broker sla-accords-provisioning
```

Upon successful completion an individual virtual machine will have been provisioned and deployed for each of the provisioning interface components and interconnected to operate around the shared common publisher instance. Please note that deployment will be for the account specified during the creation of the collection of manifests and associated service level agreement.

The resulting large model Accords Platform is now ready for use for the description, provisioning and deployment of subsequent resources. In order to do so the following command lines may be used for parsing and brokering of a test manifest "*manifest.xml*" through this composite platform:

```
$ export publisher="large platform publisher url"
$ testcp –tls security/testcpTls.xml –publisher $publisher manifest.xml
$ testcb –tls security/testcbTls.xml –publisher $publisher plan_manifest.xml
```

The first line of the above sequence requires the definition of the full URL providing access to the publisher of the Large Model Accords Platform. The security information provided by the "tls" option may be omitted if insecure operation was required for the deployment of the large model platform.

Elasticity and Scaling

The above described scenario whereby the deployment of the Large Accords Platform is performed and monitored by the Small Accords Platform, itself under the monitoring control of a NAGIOS platform, may easily be extended to allow for full scalability. This can be achieved by the addition of a CompatibleOne Optimal Loading, COOL, component node to each of the Large Model manifests and providing the appropriate configuration instructions. An example of this type of manifest is shown below for the COMONS module. The same approach may be taken for other, even all, component manifests as required to satisfy the desired scalability of the overall platform.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest name="accords:comons" xmlns="http://www.compatibleone.fr/schemes/manifest.xsd">
        <node name="accords:publisher" access="public"
                scope="common" type="accords:publisher"/>
        <node name="comons" access="public" scope="normal"
                type="simple" provider="openstack">
                <infrastructure name="accords:component"/>
                <image name="debian:comons">
                        <system name="debian_with_cosacs"/>
                </image>
        </node>
        <node name="cool" type="simple" access="public" scope="normal" provider="openstack">
        <infrastructure name=" accords:component "/>
        <image name=" debian:cool">
        <system name="debian_with_cosacs"/>
        </image>
        </node>
<configuration name="debian:comons">
        <action expression="comons.configure(accords:publisher.publisher.hostname);"/>
        <action expression="comons.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-comons');"/>
        <action expression="comons.fork('bash run-comons');"/>
        <action expression="cool.system('wget http://www.compatibleone.fr/accords-
platform/debian/run-cool');"/>
        <action expression="cool.configure(comons.contract);"/>
        <action expression="cool.system('export elastic_contract=$comons_contract');"/>
        <action expression="cool.system('export elastic_ceiling=7');"/>
        <action expression="cool.system('export elastic_floor=4');"/>
        <action expression="cool.fork('bash run-cool 8087');"/>
</configuration>
<release name="debian:comons">
        <action expression="comons.kill('comons');"/>
        <action expression="comons.system('sleep 2');"/>
</release>
<security name="accords:security"/>
<account name="accords:account"/>
</manifest>
```

The sections in bold blue text have been added to provide scalability. The load balancing node, “cool”, is configured to receive the contract identifier of the workload node that is to be scaled out. The static scalability conditions are here expressed by the floor and ceiling values indicating that at least four additional contracts will be initially required for anticipated load management and that this can expand up to a total of seven additional contracts in order to be able to handle peak conditions as and when they are encountered. The load balancer is launched indicating the HTTP service port on which it is to LISTEN for incoming connections. Further work will be necessary in this field for the ad-equation of the load balancing approach for other protocols and scalability algorithms.

Service Level Agreements

The Accords Platform makes provision for operation under the control of two distinct types of service level agreements:

1. Provider Service Level Agreements
2. Customer Service Level Agreements

In both cases the agreement document type, encapsulating the service description terms, is based on an international standard known as WS-Agreement. This document format makes provision for the description of the agreement in the following terms:

- The parties to the agreement
- The duration of the agreement
- The agreement template
- The service Description
- The service conditions
- The service guarantees

The service description section of the agreement format is provided in the form of a CompatibleOne manifest document. The service conditions describe the eventual placement conditions and the service guarantees describe the monitoring requirements and eventual business value consequences.

Provider Service Level Agreement

The provider service level agreement describes and establishes the conditions under which placement will occur for the particular provider. This is defined in terms of the collection of quota records that are made available by the provider for use by the CompatibleOne Placement Services. During the service instance creation phase of the brokering operation a request for placement will be sent to the Placement Services for the creation of a “provision”. The request will provide the description of the required resources by providing the manifest “node” reference. This information will be used to locate the collection of suitable providers for which the requested technical characteristics are currently available, as described by their provider service level agreement defined quota. The placement services will reserve the quota elements as required with the appropriate quantities being appended to the resulting placement description. Upon completion of the service graph creation phase the broker will commence the deployment phase whereby placement quantities will be converted from their initial reserved state to the consumed state reflecting the use of the corresponding quota described resources.

The service description section of the WS-Agreement framework of a Provider Service Level Agreement makes reference to a Provider element defined in an external manifest and describing the corresponding quota information. The Agreement header references the account of the operator

of the provider interface for accountancy purposes. An example of the Provider Agreement is shown below for the OpenStack PROCCI interface of the Small Accords Platform.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<agreement
        xmlns="http://www.compatibleone.fr/schemes/slam.xsd"
        name="osprocci" initiator="osprocci" responder="accords"
        serviceprovider="initiator"
        description="generated provider SLA"
        initiation="now" expiration="never" template="none">
        <terms name="osprocci:services" type="services">
                <term name="osprocci:service1">
                        <provider name="osprocci”/>
                </term>
        </terms>
        <terms name="osprocci:conditions" type="conditions">
                <term name="osprocci:condition1">
                        <variable name="osprocci:v1"
                                property="occi.placement.algorithm"
                                condition="eq" value="zone"/>
                        <variable name="osprocci:v2"
                                property="occi.placement.zone"
                                condition="eq" value="europe"/>
                </term>
        </terms>
        <terms name="accords:guarantees" type="guarantees"/>
</agreement>
```

The above agreement defines the service description terms to be described in the provider instance named “osprocci”. The placement conditions require the geographical location to be “Europe”. No service guarantees are required.

The following XML document shows the provider information and associated quota:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<provider
        xmlns="http://www.compatibleone.fr/schemes/provision.xsd"
        name="osprocci" operator="osprocci" zone="europe">
        <quota name=":cpu" property="occi.compute.cores" offered="30">
                <price name=":cpu:price" operator="osprocci"
                        rate="0.04" currency="euro" period="hour"/>
        </quota>
        <quota name=":ram" property="occi.compute.memory" offered="30">
                <price name=":ram:price" operator="osprocci"
                        rate="0.01" currency="euro" period="hour"/>
        </quota>
        <quota name=":disk" property="occi.storage.size" offered="1000">
                <price name=":disk:price" operator="osprocci"
                        rate="0.03" currency="euro" period="hour"/>
        </quota>
        <quota name=":vm" property="accords.vm.count" offered="30">
                <price name=":vm:price" operator="osprocci"
                        rate="1.00" currency="euro" fixed="true"/>
        </quota>
        <quota name=":adr" property="accords.address.count" offered="30">
                <price name=":adr:price" operator="osprocci"
                rate="0.05" currency="euro" period="hour"/>
        </quota>
</provider>
```

When stringent security conditions have been activated these two documents must have been parsed into the Accords Platform OCCI category services prior to the activation of the corresponding operator’s PROCCI interface otherwise the startup of the provisioning interface will be refused.

Customer Service Level Agreement

The customer service level agreement describes and establishes the conditions under which the provisioning of service must occur for a particular service and for a particular customer of the corresponding Accords Platform operator. The service level agreement describes the service to be deployed in terms of a CompatibleOne Manifest. The service conditions section allows description of placement conditions and the guarantee section allows definition of conditions that require monitoring and eventual business values that they represent.

An XML document showing a simple example of a customer service level agreement can be seen below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<agreement
        xmlns="http://www.compatibleone.fr/schemes/slam.xsd"
        name="xwiki-cassandra"
        initiator="jamie" responder="accords" serviceprovider="responder"
        description="generated SLA" initiation="now" expiration="never" template="none">
        <terms name="xwiki-cassandra:services" type="services">
                <term name="xwiki-cassandra:service1">
                        <manifest name="xwiki-cassandra"/>
                </term>
        </terms>
        <terms name="xwiki-cassandra:conditions" type="conditions">
                <term name="xwiki-cassandra:condition1">
                        <variable name="xwiki-cassandra:v1"
                                property="occi.placement.algorithm"
                                condition="eq" value="quota:zone"/>
                        <variable name="xwiki-cassandra:v2"
                                property="occi.placement.zone"
                                condition="eq" value="europe"/>
                </term>
        </terms>
        <terms name="accords:guarantees" type="guarantees"/>
</agreement>
```

In the above agreement the service description section refers to the POC1V3 manifest description of the deployment of the XWIKI Application Server using the Cassandra Replicated Database Engine The placement conditions require that deployment be performed in Europe. No service guarantees are defined.

The following example shows a service level agreement that could be used for the deployment of the large Accords Platform and in this case defines a single, security oriented guarantee clause requiring that the maximum allowed logged in user count not exceed 1 user.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<agreement
        xmlns="http://www.compatibleone.fr/schemes/slam.xsd"
        name="accords-platform"
        initiator="jamie" responder="accords" serviceprovider="responder"
        description="generated SLA"
        initiation="now" expiration="never" template="none">
        <terms name=":services" type="services">
                <term name=":service1">
                        <manifest name="accords:platform"/>
                </term>
        </terms>
        <terms name=":conditions" type="conditions">
                <term name=":condition1">
                        <variable name=":v1"
                                property="occi.placement.algorithm"
                                condition="eq" value="quota:zone"/>
                        <variable name=":v2"
                                property="occi.placement.zone"
                                condition="eq" value="europe"/>
                </term>
```

```
        </terms>
        <terms name=":guarantees" type="guarantees">
                <term name=":guarantee3">
                        <guarantee name=":g3" obligated="provider"
                                importance="normal" scope="default">
                                <variable name=":gv3"
                                        property="user:count" condition="notgreater" value="1"/>
                                        <business name=":bv3" nature="reward"
                                                expression="cordscript:packet.consume();"/>
                                        <business name=":bv3" nature="penalty"
                                                expression="cordscript:alert.new();"/>
                        </guarantee>
                </term>
        </terms>
</agreement>
```

Further information and examples concerning service level agreements may be found in the accompanying document “CompatibleOne Resource Description System Technical Reference V1.12”.

Monitoring

The enforcement of service level agreement guarantees, as described in the preceding section relating to Customer Service Level Agreements, requires complex and integrated monitoring mechanisms to be deployed and managed by the Accords Platform. The accompanying document "CompatibleOne Monitoring Service" is devoted to the description of the subject.

Financial

Until now, other than the presence of pricing information described for the quota of the provider descriptions required for the provisioning interface Provider Service Level Agreements, no mention has been made concerning pricing, costing, billing and other financial concerns and operations.

The underlying OCCI category instance management libraries, on which the Accords Platform Access Protocol is built, offer the possibility of associating a price with the creation of instances of a particular category. This allows different commercial policies to be determined and developed by the pricing of different fundamental elements of the Accords Platform. Since the commercial operation of the Accords Platform, by a business operator, will naturally incur the costs of deployment of the resources required for the overall offer of service, it is only normal that these costs be covered in some way or other. This is true especially in the case of an offer based on the fully scalable version of the platform. Considering a case initially requiring the deployment of twenty virtual machines for the complete offer of service, with an anticipated initial load requiring an elastic floor of ten machines and an elastic ceiling of 100 machines the overall cost of operation would be calculated as follows:

Platform Cost	20 * "unit machine price"
Elastic Floor	10 * "Platform Cost Price"
Elastic Ceiling	100 * "Platform Cost Price"

With a realistic, current market, unit machine price of around only 0.40€ per hour this would give an overall operational price ranging between the simple Large Platform deployment value of 8€ (0.40€ * 20), the elastic floor value of 80€ (0.40€ * 20 * 10) and the elastic ceiling value of 800€ (0.40€ * 20 * 100) and this for each hour of operation.

It is important that these costs, equally in terms of compute, network and storage, be spread evenly across the pricing schemes in place for the customers of the platform in a proportional manner.

The pricing of individual and fundamental category instance creation allows just this. Price information detected for a particular category at the moment of creation of an instance, of say a contract management record, would generate a financial transaction of the corresponding amount for the corresponding customer account. The eventual deletion of the same instance would generate the corresponding terminal transaction and consequently allowing the price of ownership of this piece of information, processing and storage to be evaluated.

Whilst the account, price, transaction and invoice category management system has been built into the heart of the Accords Platform models, small, large and scalable, the business intelligence required for the determination and enforcement of such a commercial policy management system is well outside of the scope of this current work.

Conclusion

The Accords Platform provides a comprehensive solution for the flexible and extensible description and management of cloud services and resources. The proposed three deployment models of Small, Large and Scalable are sufficiently flexible for the satisfaction of the different requirements of a diverse range of commercial applications.

The domain of Cloud Computing being in a state of constant change, the adoption of the Open Cloud Computing Interface OCCI, for the interfaces of the Accords Platform, provides the basis on which interoperability can be built allowing extension of the model to meet future market demand.

Further work can easily be imagined and be seen to be necessary in particular in the more commercial areas concerning “Service Level Agreements”, their “Negotiation” and impact of “Business Values” upon “Commercial Policy” it’s “Management” and “Enforcement”.

Iain James Marshall

CTO CompatibleOne

Prologue