

Amenesik

# Platform Editor

User Guide Version 1.0a

Iain James Marshall  
19/06/2016

## Table des matières

Introduction .....	4
Overview .....	4
User Interface .....	5
The Platform Editor .....	6
Create Platform .....	6
Select Platform .....	7
Load Platform .....	7
Inspect Platform .....	7
Nature .....	7
System .....	8
Database .....	8
Interface .....	8
Compute .....	8
Storage .....	8
Volume .....	8
Delete Platform .....	8
Copy Platform .....	8
Merge Platform .....	9
Drop Platform .....	9
Select Component .....	9
Add Component .....	9
Drop Component .....	9
Generate Platform .....	9
Platform XML .....	9
Platform Production .....	10
Manifest XML .....	10
The Component Editor .....	12
Create Component .....	12
Select Component .....	12
Delete Component .....	13
Load Component .....	13
Inspect Component .....	13
Nature .....	13
Consumer .....	13
Authorize .....	14

Compute.....	14
Storage .....	14
Copy Component .....	14
Merge Component.....	14
Drop Component .....	14
Select Category .....	14
Add Category.....	14
Drop Category .....	14
The Category Editor .....	15
Create Category .....	15
Select Category .....	16
Load Category .....	16
Inspect Category .....	16
Name.....	16
Location.....	16
Access.....	16
Operator.....	17
Delete Category .....	17
Copy Category.....	17
Merge Category .....	17
Drop Category .....	17
Create Member.....	17
Select Member.....	17
Inspect Member.....	18
Name.....	18
Required.....	18
Immutable.....	18
Is Index .....	18
Type.....	18
Value .....	19
Dimension .....	19
Delete Member.....	19
Add Member .....	19
Drop Member .....	19
Create Action .....	19
Select Action .....	19

Inspect Action .....	19
Name .....	20
Scheme .....	20
Type .....	20
Body .....	20
Delete Action .....	20
Add Action .....	20
Drop Action .....	21
References .....	22
OCCI .....	22
TOSCA .....	22
CIMI .....	22
CORDS .....	22
AMENESIK .....	22

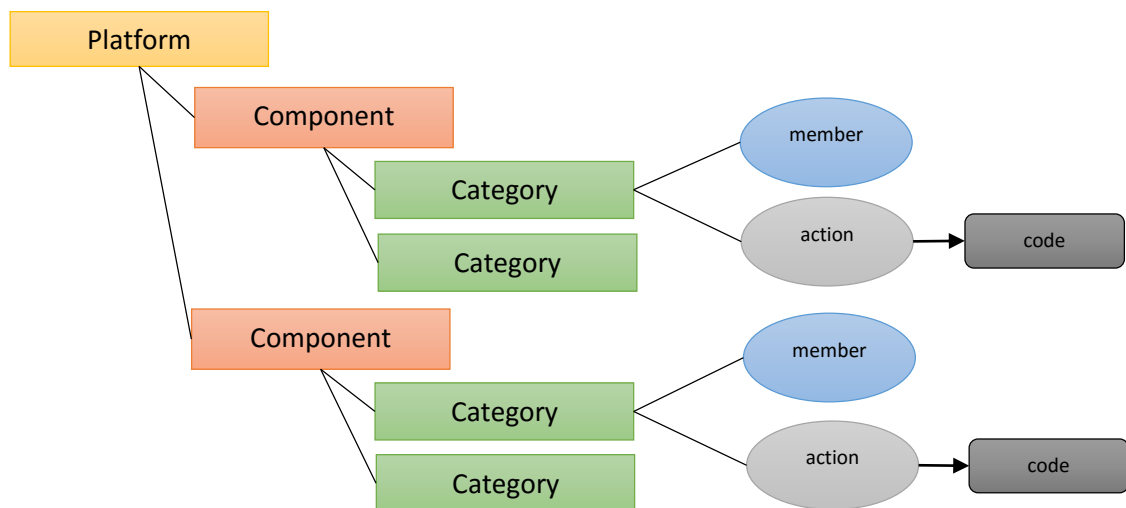
## Introduction

This document describes the operations and behaviour of the new Amenesik Platform Editor designed, developed and distributed by Amenesik, for the definition, deployment and management of Amenesik Cloud Engine industrialised version instances of the Accords Platform Cloud Brokerage and Cloud Provisioning software initially developed under Apache 2 license during the CompatibleOne project.

## Overview

The Amenesik Platform Editor is an integrated tool allowing the definition and production of platform models in terms of operational components and the subsequent definition of these components in terms of collections of categories. The tool also allows for the definition of OCCl categories in terms of their member attributes and their action methods including specification and management of the code implementing their required behaviour.

The following diagram shows the Platform Object Model:

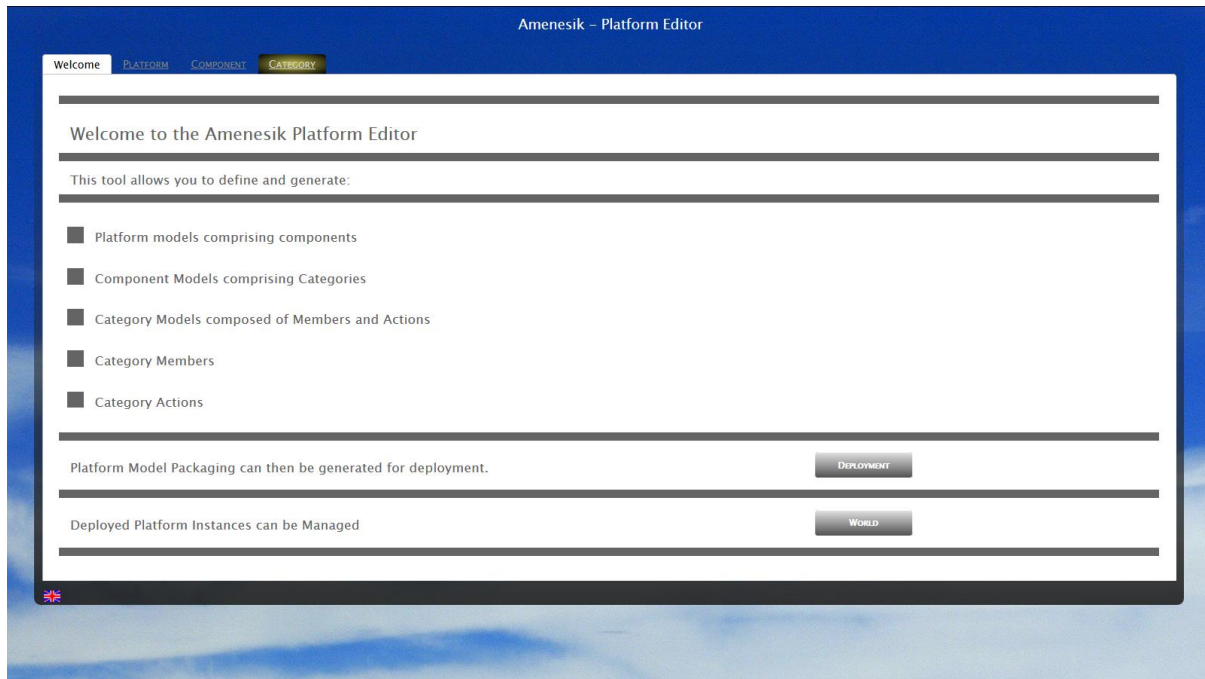


From the above diagram the following points should be understood:

- A Platform is composed of one or more Components
- A Component is composed of one or more Categories
- A Category is composed of one or more Members and Actions
- A block of code can be defined for each Action.

## User Interface

The following screen shot shows the user interface of the entry point to the Amenesik Platform Editor that is accessible through the URL <http://www.amenesik.com/cobe/cobuild.php> once that the successful authentication of the user identification has been performed.



The user interface comprises the following tab pages:

1. The Welcome page
2. The Platform Editor page
3. The Component Editor page
4. The Category Editor page

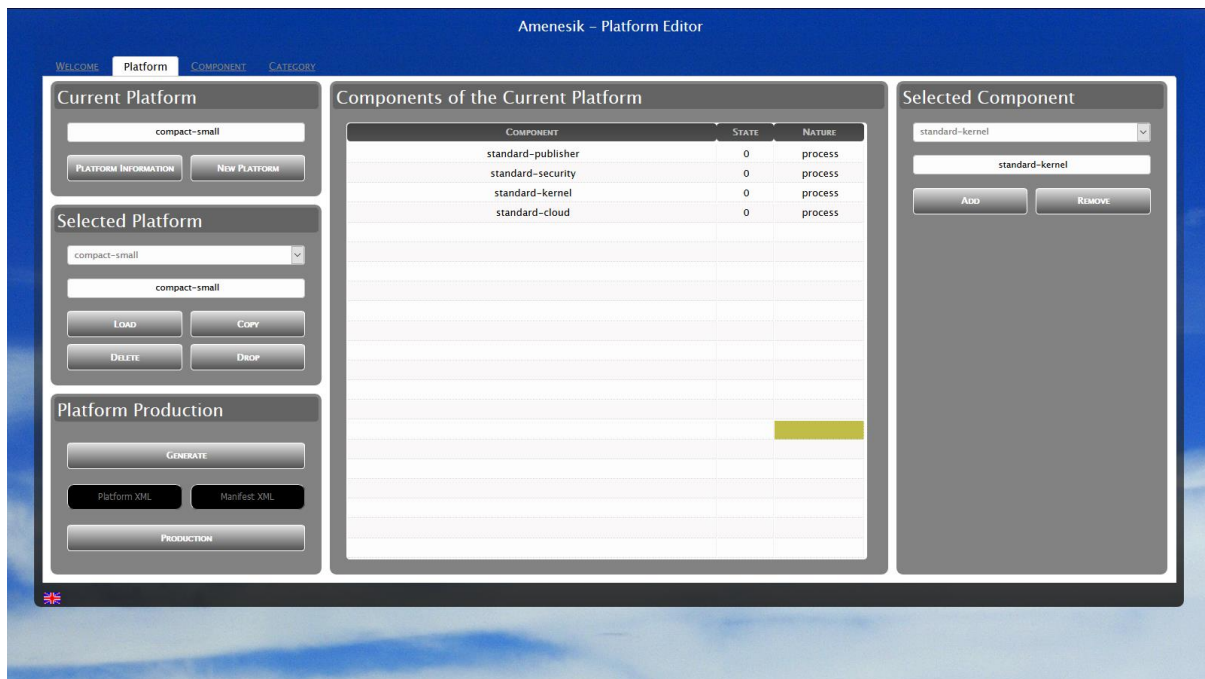
Each of the individual editor pages comprise similar functional regions and an overall operational approach that will be presented in detail in the following sections of this document.

The “Welcome” page offers access to the Platform Deployment tool for the preparation of the Service Level Agreement required for the deployment of a Platform Model prepared by the Amenesik Platform Editor Tool.

Access to the Amenesik Platform Service deployment management tool is also available from the Welcome page for the management of Amenesik Cloud Engine platform deployments.

## The Platform Editor

The following screen shot shows the layout of the Platform Editor page:



The Platform Editor comprises the following regions:

1. The Current Platform region shows the name of the platform currently loaded into the platform editor. The region also offers a push button for inspection of the platform properties dialog box and for the creation of new platform definitions.
2. The Platform Selection region offers a drop down list of the names of all currently defined platforms. The region shows the name of a platform that has been selected from this list along with the push button controlled operations that may be performed on this selected platform.
3. The Platform Component List region shows the list of components that comprise the current platform model under definition. By clicking on a component name in this list, the corresponding component definition will be loaded and user interface control will be transferred to the Component Editor page.
4. The Component Selection region offers a drop down list of the names of all currently defined components. The region shows the name of the currently selected component and the collection of push button controlled operations that may be performed on this selected component.

The Platform Editor page also offers a collection of push buttons allowing generation of the platform description resource and launch of the platform production and packaging tool. Further push buttons allow inspection of the platform description XML resource file and the corresponding platform deployment manifest.

### Create Platform

A new platform model can be created by clicking on the “New Platform” push button of the Platform Editor page. A new, empty, platform model definition will be created and the Platform Model properties dialog box will be presented allowing the platform model to be defined. The model will then be loaded as the current platform model in the editor.

## Select Platform

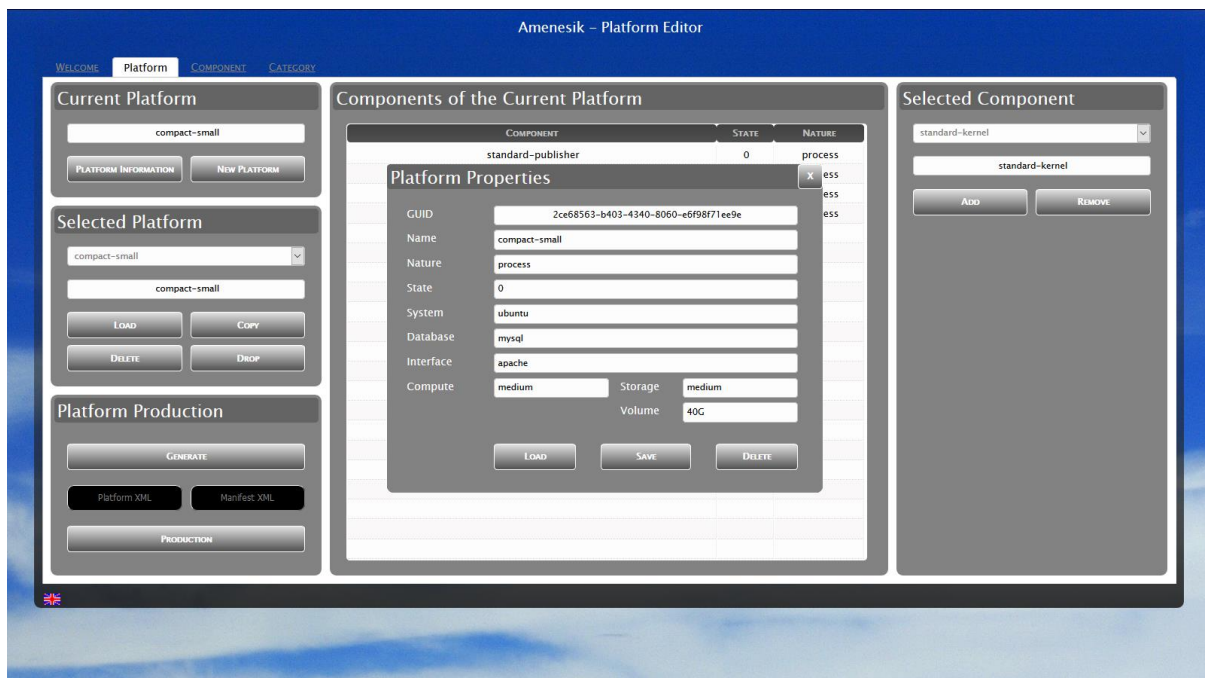
A platform can be selected from the drop down list of platform names on the Platform Editor page. The name of the selected platform model will appear in the associated display field and the push buttons of the operations that can be performed on the selected platform will be displayed.

## Load Platform

This button will be displayed when a platform has been selected and when actioned it will load the selected platform model as the current model in the Platform Editor. The Platform Component List will show the list of components of the loaded platform model.

## Inspect Platform

This button will be displayed when a platform model has been loaded by the preceding *Load Platform* operation and when actioned will present the Platform Properties Dialog Box for consultation and modification of the properties associated with the platform. The following screen capture shows the Platform Properties Dialog Box:



The nature and behaviour of a Platform Model is defined by the Platform Properties as described below.

## Nature

This property defines the deployment nature of the platform and directly influences the architecture in terms of the number of virtual machines deployed and the relationships between them. This property should take one of the following values.

### PROCESS

The components of Platforms of this nature will be deployed as processes of a single virtual machine along with their database and user interface support components.

### MACHINE

Each of the components of Platforms of this nature will be deployed as individual virtual machine including their database and user interface support components. Each virtual machine will be configured to communicate with the other components as required to perform their function.



## CLUSTER

Each of the components of Platforms of this nature will be deployed as clusters of virtual machines operating around a central database system, in its own machine, with a load balanced user interface management cluster. Component clusters will be configured to communicate with the other load balancing endpoints of the other component clusters as required to perform their function.

## System

This property allows the base operating system to be defined for the deployment of the virtual machines and clusters. This should be set to “Ubuntu” since this is currently the only operating system on which the Platform deployment has currently been qualified.

## Database

This property allows the nature of the database engine used for the storage of all category instances to be defined for use by the Amenesik Cloud Engine. This should be set to “mysql” since this is the only database engine that has been qualified.

## Interface

This property allows the web server used for access to the Amenesik Cloud Engine to be defined. This should be set to “apache” since this is the only web application server engine that has been qualified.

## Compute

This property allows the name of the compute definition to be specified as “small”, “medium”, “large” or other values. The actual compute instances defined by these names can be defined by the usual means and is outside of the scope of this document.

## Storage

This property allows the name of the storage definition to be specified as “small”, “medium”, “large” or other values. The actual storage instances defined by these names can be defined by the usual means and is outside of the scope of this document.

## Volume

This property allows the size of a persistent storage volume to be specified in GB (gigabytes). This is not mandatory but highly recommended for the safety of the accords platform database. When a persistent storage volume has been defined then the database and all long term information will be stored on that volume. In case of an instance reboot with a new platform instance being built the same storage volume will be attached and the database contents will be preserved. Without a persistent volume the contents of the database will be lost.

## Delete Platform

This button will be displayed when a platform has been selected and when actioned it will delete the selected platform model and any links to components that may have been defined for the platform model.

## Copy Platform

This button will be displayed when a platform has been selected and when actioned it will remove all components from the platform being edited and then add links to components that are defined for the selected platform. This operation has the effect of *duplicating* a platform model under a different name and with different platform model properties.

### Merge Platform

This button will be displayed when a platform has been selected and when actioned it will add links to components that are defined for the selected platform to the platform model currently being edited. This operation has the effect of *adding* a collection or group of components from a platform model.

### Drop Platform

This button will be displayed when a platform has been selected and when actioned it will delete all links to components that are defined for the selected platform from the platform model currently being edited. This operation has the effect of *removing* a collection or group of components from a platform model.

### Select Component

A component can be selected from the drop down list of component names on the Platform Editor page. The name of the selected component model will appear in the associated display field and the push buttons of the operations that can be performed on the selected component will be displayed.

### Add Component

This push button will appear when a component has been selected by the preceding *Component Selection* operation and when actioned will add a link to the selected component to the current platform model. The component will appear in the Platform Component List.

### Drop Component

This push button will appear when a component has been selected by the preceding *Component Selection* operation and when actioned will remove all links to the selected component from the current platform model. The Platform Component List will be refreshed.

### Generate Platform

This push button will generate the XML Platform resource description corresponding to the platform model currently loaded into the Platform Editor. This XML file will comprise a single platform element comprising all the nested component and categories elements referenced by the Platform Model.

### Platform XML

This push button allows inspection of the XML Platform resource description produced as a result of the previous operation *Generate Platform*. The XML file will be retrieved and displayed in a new tab page of the current browser.

The following screen capture shows an example of the display of the XML file:

```

- <platform name="compact-small" nature="process" system="ubuntu" interface="apache" database="mysql" volume="40G">
- <component name="standard-publisher" authorize="no" consumer="">
- <category term="publication" access="private" operator="none">
- <attributes>
- <attribute name="remote" type="string"/>
- <attribute name="what" type="string"/>
- <attribute name="region" type="string"/>
- <attribute name="why" type="string"/>
- <attribute name="stamp" type="int"/>
- <attribute name="uptime" type="int"/>
- <attribute name="who" type="string"/>
- <attribute name="pass" type="string"/>
- <attribute name="identity" type="string"/>
- <attribute name="zone" type="string"/>
- <attribute name="price" type="string"/>
- <attribute name="rating" type="string"/>
- <attribute name="operator" type="string"/>
- <attribute name="pid" type="int"/>
- <attribute name="state" type="int"/>
- </attributes>
- <actions>
- <action id="DELETE"/>
- <action id="suspend"/>
- <action id="restart"/>
- </actions>
- </category>
- <category term="enquiry" access="private" operator="none">
- <attributes>
- <attribute name="region" type="string"/>
- <attribute name="what" type="string"/>
- <attribute name="who" type="string"/>
- <attribute name="pass" type="string"/>
- <attribute name="state" type="int"/>

```

## Platform Production

This push button will launch the COMODEL Platform Production tool that will process the XML Platform resource file.

This processing will perform the following list of actions:

1. Generation of the Component Sources
2. Generation of the Category Sources
3. Generation of the Action Sources
4. Generation of all Component Deployment Scripts
5. Generation of all Component Installation Scripts
6. Generation of all Component Launch Scripts
7. Generation of all Component Maintenance Scripts
8. Compilation of all Component Source
9. Linking of all Component Objects
10. Packaging of all Component Packages
11. Delivery of all Component Deployment Packages to the Depot
12. Generation of the Platform Deployment Manifest

The Source Generation will respect the following directory structure:

- Component sources, both C and H files, will be generated to the directory of the same name as the Component Model under the parent directory */home/c4/standard-components/*.
- Category sources, both C and H files, will be generated to the directory of the same name as the Category Model under the parent directory */home/c4/standard-categories/*.
- All other source files, XML and scripts will be generated into the of the same name as the Platform Model under the parent directory */home/c4/standard-platforms/*.
- All deployment scripts and deployment packages will be generated into the directory of the same name as the Platform Model under the parent directory */home/c4/standard-depot/*.

## Manifest XML

This push button allows inspection of the XML Manifest document produced as a result of the previous operation *Platform Production*. The XML file will be retrieved and displayed in a new tab page of the current browser.

This composition of this deployment manifest will depend on the nature of the Platform Production Model. Three basic types of Platform Production Model are possible:

1. Process

All components of this type of Platform Production Model will be generated as processes of a single node and its resulting deployed virtual machine. The configuration action expressions of the manifest will all target the same node for the installation of the software packages of the components defined for a particular Platform Model.

2. Machine

All components of this type of Platform Production Model will be generated as individual nodes and their resulting deployed virtual machines. The configuration action expressions of the manifest will target the specific node for the installation of the software packages of the corresponding components a particular Platform Model.

3. Cluster

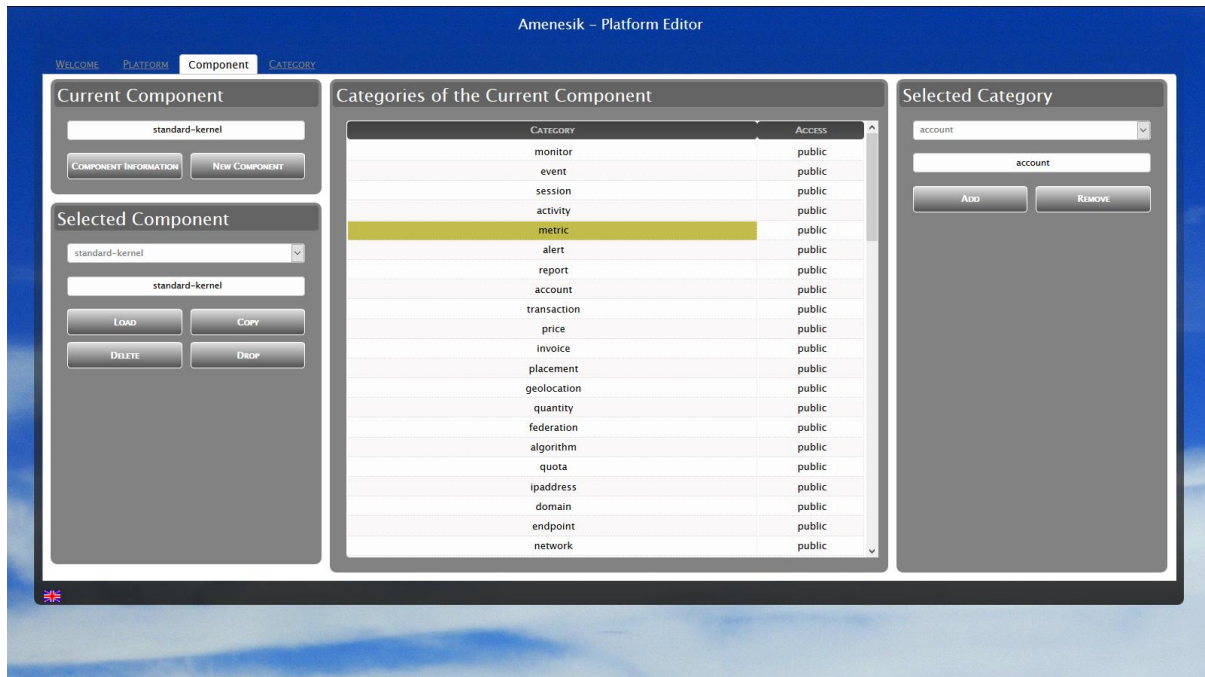
All components of this type of Platform Production Model will be generated as individual load balanced clusters of nodes and their resulting deployed virtual machines. The configuration action expressions of the manifest will target the specific node clusters for the installation of the software packages of the corresponding components a particular Platform Model.

The deployment Manifest document will not be parsed into the Amenesik Cloud Engine. This must be performed using either the Command Parser or the Remote Command Parser tools before deployment of the platform may occur. This is outside of the scope of this tool and this documentation. The following screen capture shows an example of an XML manifest:

```
- <manifest name="compact-small">
- <node name="compact-small" entry="http" access="public" type="simple" provider="any">
- <infrastructure name="compact-small">
- <compute name="compact-small" cores="1" speed="1G" memory="4G" architecture="x86_64"/>
- <storage name="compact-small" size="10G">
- <volume name="compact-small" size="40G" mount="sdf" label="default" image="none" type="persistent"/>
- </storage>
- <network name="compact-small" label="account">
- <port name="cosacs" protocol="tcp" from="8286" to="8286" range="0.0.0.0/24"/>
- <port name="ssh" protocol="tcp" from="22" to="22" range="0.0.0.0/24"/>
- <port name="http" protocol="tcp" from="80" to="80" range="0.0.0.0/24"/>
- <port name="https" protocol="tcp" from="443" to="443" range="0.0.0.0/24"/>
- <port name="command" protocol="tcp" from="8186" to="8186" range="0.0.0.0/24"/>
- <port name="component0" protocol="tcp" from="8086" to="8086" range="0.0.0.0/24"/>
- <port name="component1" protocol="tcp" from="8087" to="8087" range="0.0.0.0/24"/>
- <port name="component2" protocol="tcp" from="8088" to="8088" range="0.0.0.0/24"/>
- <port name="component3" protocol="tcp" from="8089" to="8089" range="0.0.0.0/24"/>
- </network>
- </infrastructure>
- <image name="compact-small" agent="cosacs">
- <system name="ubuntu-with-cosacs"/>
- </image>
- </node>
- <configuration name="compact-small">
- <action name="c_compact-small1" expression="compact-small.system('wget http://cloud.amenesik.com/depot/compact-small/install-host.sh');"/>
- <action name="c_compact-small2" expression="compact-small.system('./install-host.sh');"/>
- <action name="c_compact-small3" expression="compact-small.system('wget http://cloud.amenesik.com/depot/compact-small/install-drive.sh');"/>
- <action name="c_compact-small4" expression="compact-small.system('bash ./install-drive.sh xvdf sdf ext4');"/>
- <action name="c_compact-small5" expression="compact-small.system('wget http://cloud.amenesik.com/depot/compact-small/install-mysql.sh');"/>
- <action name="c_compact-small6" expression="compact-small.system('bash ./install-mysql.sh sdf');"/>
- <action name="c_compact-small7" expression="compact-small.system('wget http://cloud.amenesik.com/depot/compact-small/switch-mysql.sh');"/>
- <action name="c_compact-small8" expression="compact-small.system('bash ./switch-mysql.sh');"/>
- <action name="c_compact-small9" expression="compact-small.system('export OCCISQLHOST=127.0.0.1');"/>
```

## The Component Editor

The following screen shot shows the layout of the Component Editor page:



The Component Editor comprises the following regions:

1. The Current Component region shows the name of the component currently loaded into the component editor. The region also offers a push button for inspection of the component properties dialog box and for the creation of new component definitions.
2. The Component Selection region offers a drop down list of the names of all currently defined components. The region shows the name of a component that has been selected from this list along with the push button controlled operations that may be performed on this selected component.
3. The Component Category List region shows the list of categories that comprise the current component model under definition. By clicking on a category name in this list, the corresponding category definition will be loaded and user interface control will be transferred to the Category Editor page.
4. The region shows the name of the currently selected category and the collection of push button controlled operations that may be performed on this selected category.

### Create Component

A new component model can be created by clicking on the “New Component” push button of the Component Editor page. A new, empty, component model definition will be created and the Component Model properties dialog box will be presented allowing the component model to be defined. The model will then be loaded as the current component model in the editor.

### Select Component

A component can be selected from the drop down list of component names on the Component Editor page. The name of the selected component model will appear in the associated display field and the push buttons of the operations that can be performed on the selected component will be displayed.

## Delete Component

This button will be displayed when a component has been selected and when actioned it will delete the selected component model and any links to categories that may have been defined for the component model.

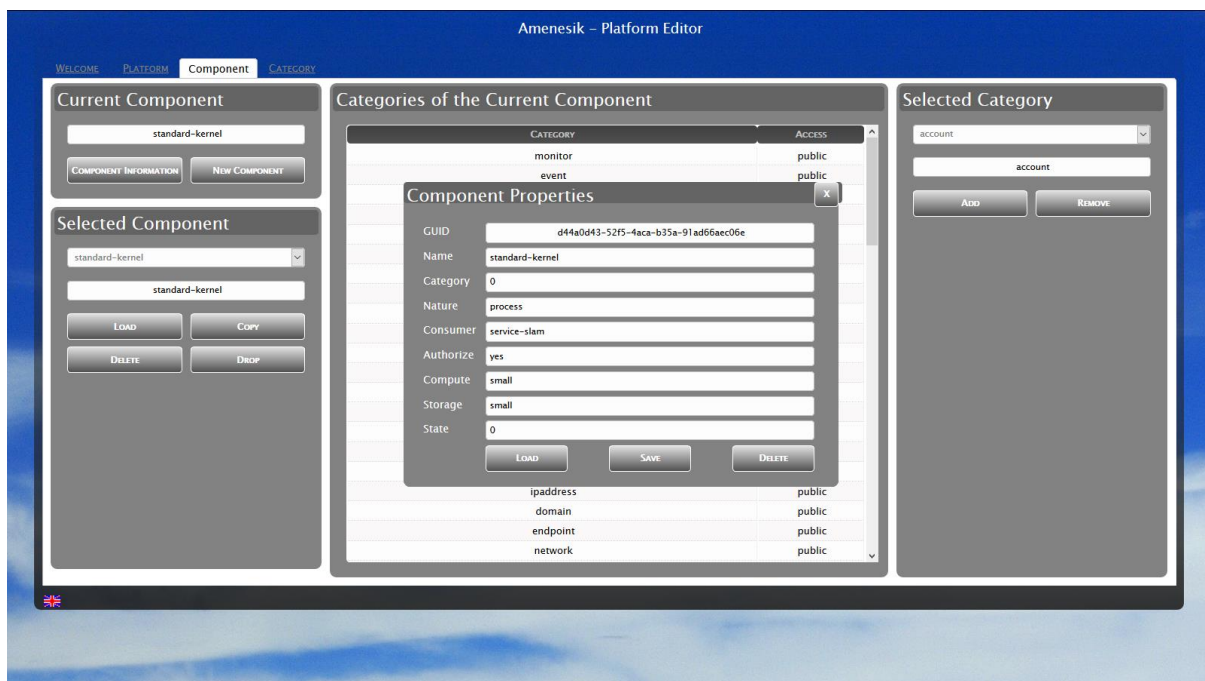
## Load Component

This button will be displayed when a component has been selected and when actioned it will load the selected component model as the current model in the Component Editor. The Component Category List will show the list of categories of the loaded component model.

## Inspect Component

This button will be displayed when a component model has been loaded by the preceding operation and when actioned will present the Component Properties Dialog Box for consultation and modification.

The following screen shot shows the Component Properties Dialog Box:



The Component Properties define the behaviour of the component and in particular its role within the Platform Operational Hierarchy. The individual fields are defined below.

### Nature

The value of this property may be used to overload the default value provided by the Platform model and would usually take one of the values described above in the section relating to the Platform Model Properties. The value of “default” indicates that the nature provided by the Platform Model is to be assumed.

### Consumer

The value of this property indicates the name of the consumer endpoint associated with this component. This is of particular importance for the connection of the monitoring components through the service level agreements manager to the deployed service contract and provisioning. The value of “service-slam” should be specified if monitoring categories are included in the component package otherwise the value of “default” or “none” should be provided.

### Authorize

When the value of this property is set to “yes” then the component is required to acquire an authorisation token during its initialisation sequence that is to be presented during all message exchanges for the identification of the endpoint and the expression of the access rights and role within the Platform model. When set to “no” authorisation may be performed but is not required of the component.

### Compute

This property may provide the name of a compute definition to override the value provided by the Platform Model Property of the same name. When set to “default” the value of the parent Platform Model will be assumed.

### Storage

This property may provide the name of a storage definition to override the value provided by the Platform Model Property of the same name. When set to “default” the value of the parent Platform Model will be assumed.

### Copy Component

This button will be displayed when a component has been selected and when actioned it will remove all categories from the component being edited and then add links to categories that are defined for the selected component. This operation has the effect of *duplicating* a component model under a different name and with different component model properties.

### Merge Component

This button will be displayed when a component has been selected and when actioned it will add links to categories that are defined for the selected component to the component model currently being edited. This operation has the effect of *adding* a collection or group of categories from a component model.

### Drop Component

This button will be displayed when a component has been selected and when actioned it will delete all links to categories that are defined for the selected component from the component model currently being edited. This operation has the effect of *removing* a collection or group of categories from a component model.

### Select Category

A category can be selected from the drop down list of category names on the Component Editor page. The name of the selected category model will appear in the associated display field and the push buttons of the operations that can be performed on the selected category will be displayed.

### Add Category

This push button will appear when a category has been selected by the preceding *Category Selection* operation and when actioned will add a link to the selected category to the current component model. The category will appear in the Component Category List.

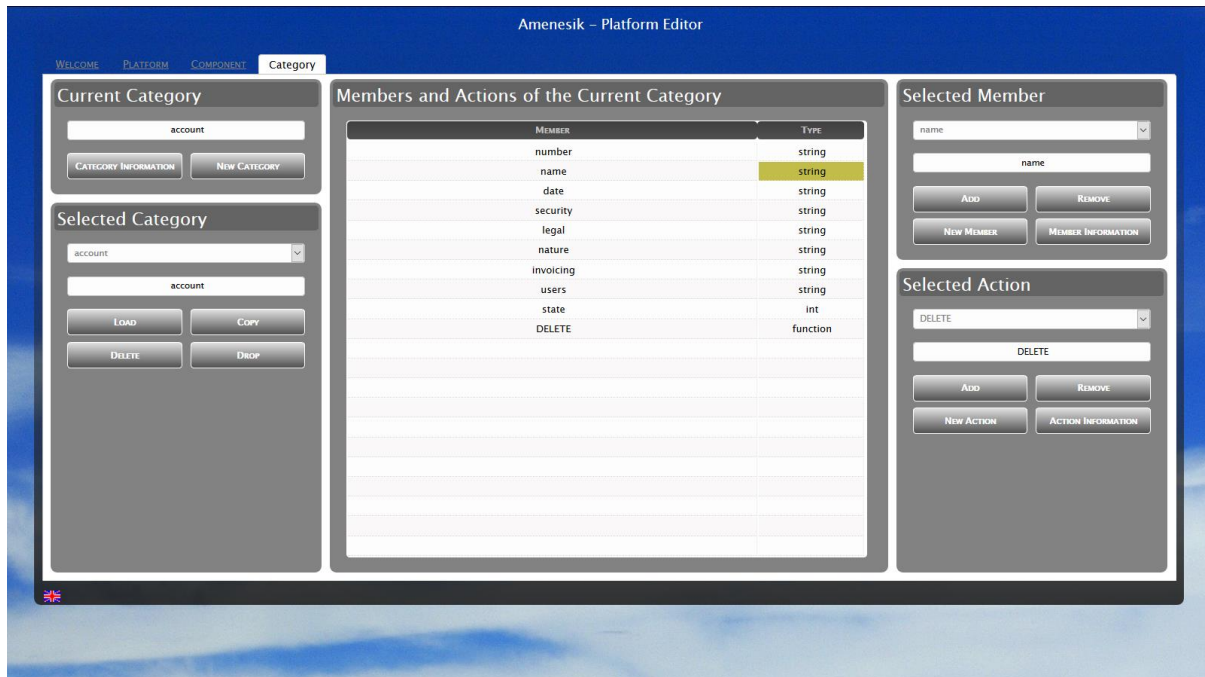
### Drop Category

This push button will appear when a category has been selected by the preceding *Category Selection* operation and when actioned will remove all links to the selected category from the current component model. The Component Category List will be refreshed.



## The Category Editor

The following screen shot shows the layout of the Category Editor page:



The Category Editor comprises the following regions:

1. The Current Category region shows the name of the category currently loaded into the category editor. The region also offers a push button for inspection of the category properties dialog box and for the creation of new category definitions.
2. The Category Selection region offers a drop down list of the names of all currently defined categories. The region shows the name of a category that has been selected from this list along with the push button controlled operations that may be performed on this selected category.
3. The Category Member and Action List region shows the list of members and actions that comprise the current category model under definition. By clicking on an action name in this list, the corresponding action definition will be loaded and user interface control will be transferred to the Action Behaviour dialog box for inspection and modification of the code of the action.
4. The Member Selection region offers a drop down list of the names of all currently defined members. The region shows the name of the currently selected member and the collection of push button controlled operations that may be performed on this selected member.
5. The Action Selection region offers a drop down list of the names of all currently defined actions. The region shows the name of the currently selected action and the collection of push button controlled operations that may be performed on this selected action.

### Create Category

A new category model can be created by clicking on the “New Category” push button of the Category Editor page. A new, empty, category model definition will be created and the Category Model properties dialog box will be presented allowing the category model to be defined. The model will then be loaded as the current category model in the editor.



## Select Category

A category can be selected from the drop down list of category names on the Category Editor page. The name of the selected category model will appear in the associated display field and the push buttons of the operations that can be performed on the selected category will be displayed.

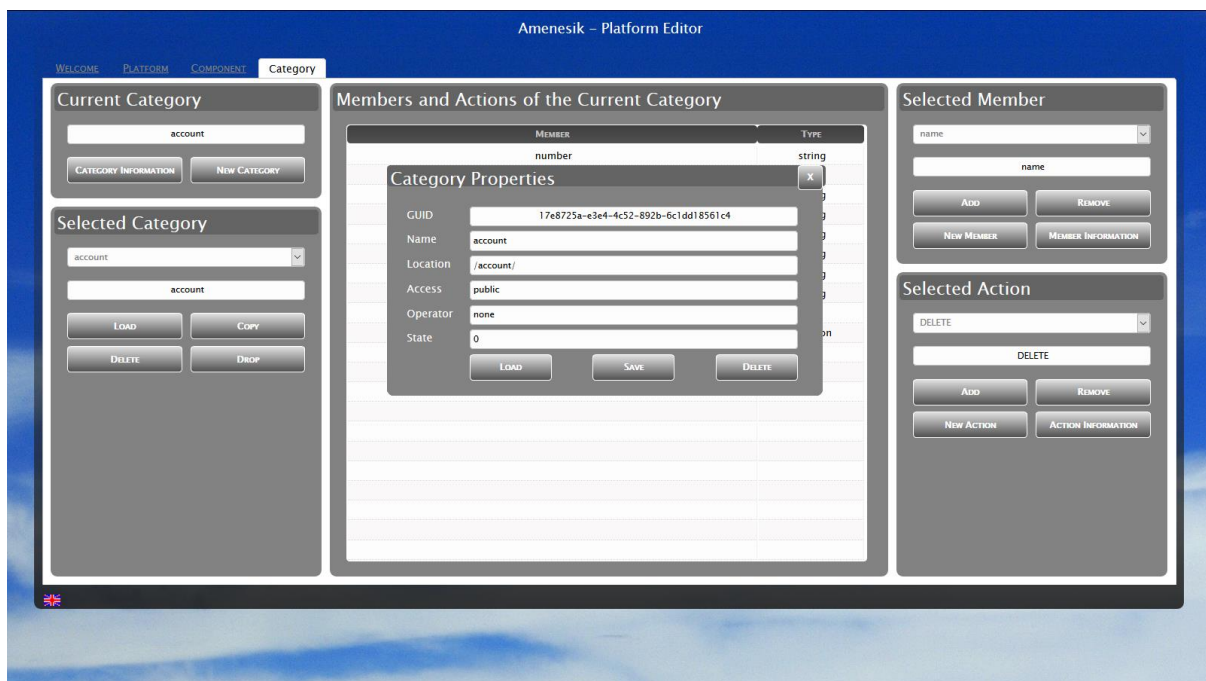
## Load Category

This button will be displayed when a category has been selected and when actioned it will load the selected category model as the current model in the Category Editor. The Category Member and Action List will show the list of members and actions of the loaded category model.

## Inspect Category

This button will be displayed when a category model has been loaded by the preceding operation and when actioned will present the Category Properties Dialog Box for consultation and modification.

The following screen shot shows the Category Properties Dialog Box:



The Category Properties provide important information concerning the category and the role that it is expected to perform within the operations of its parent component of the Platform Model.

## Name

This is the name of the OCCl category endpoint through which the category will be known.

## Location

This is the extension to the service, host name and port configuration to be added when constructing the access path URL for the OCCl category.

## Access

The value of this property indicates the intended role of the category within the operational model. This field may take one of the following values.

### PROVIDER

When set to the value of “provider” this indicates that the category represents a provisioning API encapsulation category that will be under the control of a service level agreement between the platform operator and the actual provider operator account.

### PRIVATE

When this value is set to “private” this indicates that the category is for private use by the component and is not to be published through the central service directory and publication service.

### PUBLIC

This is the default value and indicates a normal category type that is to be published through the platform publication service.

### Operator

When the access is set to the value of “provider” this field indicates the name of the operator account responsible for the provisioning endpoint.

### Delete Category

This button will be displayed when a category has been selected and when actioned it will delete the selected category model and any links to members and actions that may have been defined for the category model.

### Copy Category

This button will be displayed when a category has been selected and when actioned it will remove all members and actions from the category being edited and then add links to the members and actions that are defined for the selected category. This operation has the effect of *duplicating* a category model under a different name and with different category model properties.

### Merge Category

This button will be displayed when a category has been selected and when actioned it will add links to members and actions that are defined for the selected category to the category model currently being edited. This operation has the effect of *adding* a collection or group of members and actions from a category model.

### Drop Category

This button will be displayed when a category has been selected and when actioned it will delete all links to members and actions that are defined for the selected category from the category model currently being edited. This operation has the effect of *removing* a collection or group of members and actions from a category model.

### Create Member

A new member can be created by clicking on the “New Member” push button of the Category Editor page. A new, empty, member definition will be created and the Member properties dialog box will be presented allowing the member to be defined.

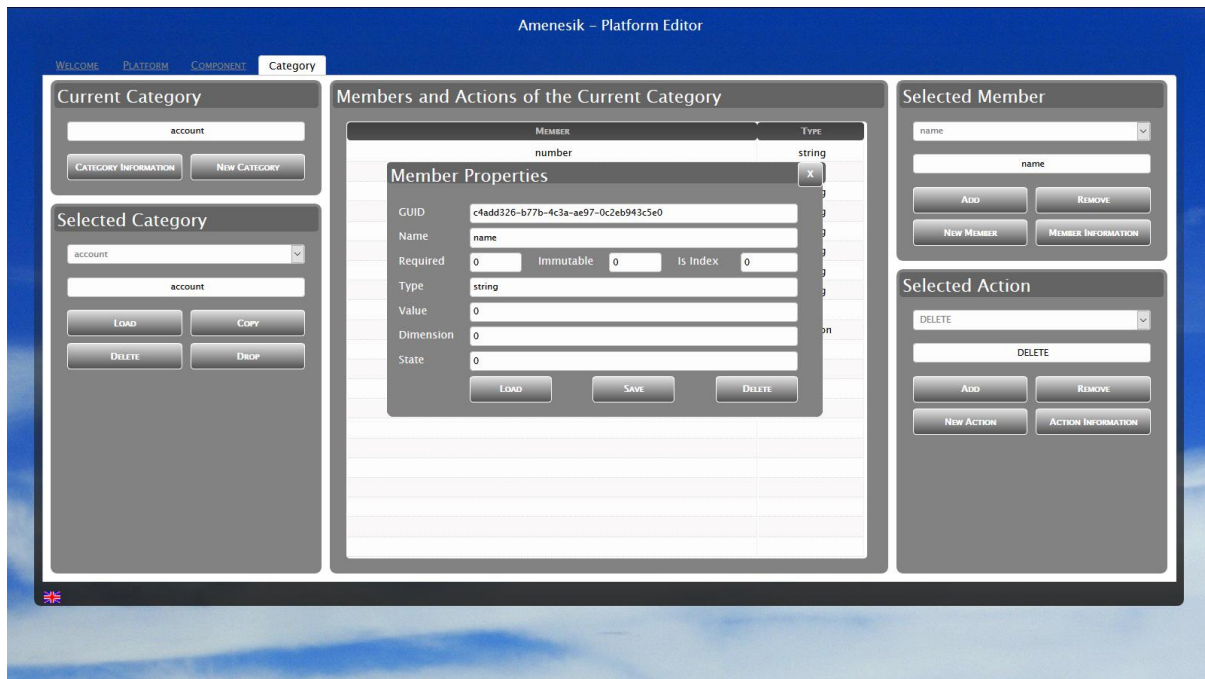
### Select Member

A member can be selected from the drop down list of member names on the Category Editor page. The name of the selected member will appear in the associated display field and the push buttons of the operations that can be performed on the selected member will be displayed.

## Inspect Member

This push button will appear when a member has been selected by the preceding *Member Selection* operation and when actioned will present the Member Properties Dialog Box for consultation and modification.

The following screen shot shows the Member Properties Dialog Box:



The Member Properties define the nature and use of the Category Member as follows.

### Name

This field provides the name of the Category Member.

### Required

When set to true this field indicates that the presence of a value for this category member is to be enforced when checking the validity of OCCl instances of this category otherwise enforcement will be relaxed.

### Immutable

When set to true this field indicates that the value for this category member cannot be set through member statements of OCCl instances received via HTTP POST or PUT methods. In this case the member is said to be read only.

### Is Index

When set to true this field indicates that the value is an index value. This is no longer used and will be phased out.

### Type

The value of this field indicates the nature of the member and may be either STRING, INTEGER or COMPONENT. In the latter case an OCCl instance URL is expected as a reference to a MEMBER object.

### Value

The value of this field provides a default value to be used when no value is provided by a member expression of an OCCl instance received via an HTTP POST for the creation of a new instance of this category.

### Dimension

The value of this field indicates if the member type is to be included amongst the type declared members of the OCCl instance category definition or is to be handled as a multiple instance un-typed linked member.

### Delete Member

This push button will appear when a member has been selected by the preceding *Member Selection* operation and when actioned will delete the currently selected member record.

### Add Member

This push button will appear when a member has been selected by the preceding *Member Selection* operation and when actioned will add a link to the selected member to the current category model. The member will appear in the Category Member and Action List.

### Drop Member

This push button will appear when a member has been selected by the preceding *Member Selection* operation and when actioned will remove all links to the selected member from the current category model. The Category Member and Action List will be refreshed.

### Create Action

A new action can be created by clicking on the “New Action” push button of the Category Editor page. A new, empty, action definition will be created and the Action properties dialog box will be presented allowing the action to be defined.

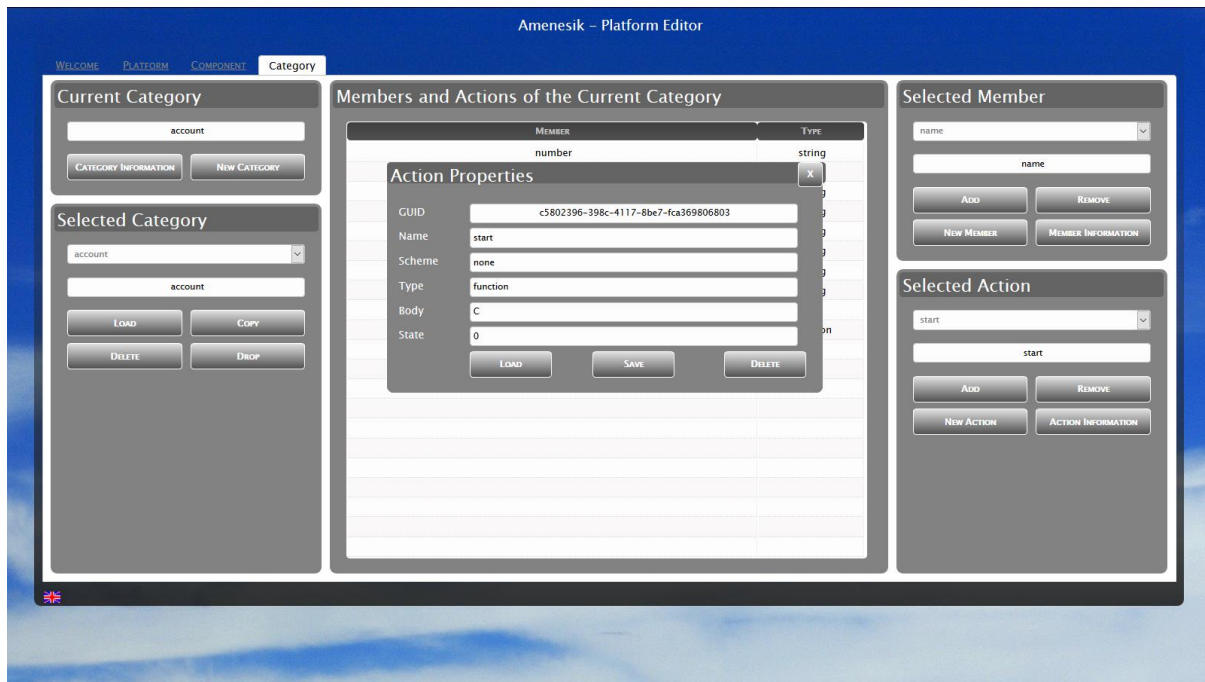
### Select Action

An action can be selected from the drop down list of action names on the Category Editor page. The name of the selected action will appear in the associated display field and the push buttons of the operations that can be performed on the selected action will be displayed.

### Inspect Action

This push button will appear when an action has been selected by the preceding *Action Selection* operation and when actioned will present the Action Properties Dialog Box for consultation and modification.

The following screen shot shows the Action Properties Dialog Box:



The Action Properties control the way in which an action is generated and the nature of its behaviour code.

### Name

This is the name of the action as expected as the value of the “?action=value” construction of the OCCl action invocation mechanism.

### Scheme

This is the value of the scheme to be used to formally define the action within the category definition.

### Type

The value of this field indicates the nature of the code block and the way in which it will be processed during code generation. The value of “C” indicates that body element contains C code to be generated as the function for the action. The value of “CORDSCRIPT” indicates that the body element contains CORDSCRIPT code to be generated and processed by the corresponding interpreter at runtime. An empty value indicates that the default mechanisms where by the presence of C code is detected during generation will be employed.

### Body

When the Type field has been set to “C” or “CORDSCRIPT” then this field will contain the corresponding code instructions to be generated during platform production.

### Delete Action

This push button will appear when an action has been selected by the preceding *Action Selection* operation and when actioned will delete the currently selected action record.

### Add Action

This push button will appear when an action has been selected by the preceding *Action Selection* operation and when actioned will add a link to the selected action to the current category model. The action will appear in the Category Member and Action List.

### Drop Action

This push button will appear when an action has been selected by the preceding *Action Selection* operation and when actioned will remove all links to the selected action from the current category model. The Category Member and Action List will be refreshed.

## References

This section of the document provides a collection of links to cloud standards documentation and Amenesik support documents.

### OCCI

The following documents are available from the OGF web site:

- OCCI CORE Version 1.1:  
<https://www.ogf.org/documents/GFD.183.pdf>
- OCCI INFRASTRUCTURE Version 1.1 :  
<https://www.ogf.org/documents/GFD.184.pdf>
- OCCI http Version 1.1 :  
<https://www.ogf.org/documents/GFD.185.pdf>

### TOSCA

The following documents are available from the OASIS web site

- TOSCA Version 1.1:  
<http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.pdf>
- TOSCA Namespace:  
<http://docs.oasis-open.org/tosca/ns/2011/12>

### CIMI

The following documents are available from the DMTF web site

- CIMI Version 1.1 :  
[http://www.dmtf.org/sites/default/files/standards/documents/DSP0263\\_1.0.1.pdf](http://www.dmtf.org/sites/default/files/standards/documents/DSP0263_1.0.1.pdf)

### CORDS

The following documents are available from the CompatibleOne community web site:

- CORDS Version 1.1 :  
<http://www.compatibleone.com/community/wp-content/uploads/2014/05/CordsReferenceManualV2.15.pdf>

### AMENESIK

The following documents are available from the AMENESIK web site:

- Amenesik Enterprise Cloud (AEC) Version 1.1:  
<http://www.amenesik.com/cloud/AmenesikCloud.pdf>
- Amenesik Cloud Engine (ACE) Version 1.1:  
<http://www.amenesik.com/cloud/AmenesikCloudEngine.pdf>
- Amenesik Manifest Editor (AME) Version 1.1:  
<http://www.amenesik.com/cloud/AmenesikManifestEditor.pdf>
- Amenesik Agreement Editor (ASE) Version 1.1:  
<http://www.amenesik.com/cloud/AmenesikServiceEditor.pdf>
- Amenesik Service Dashboard (ASD) Version 1.1:  
<http://www.amenesik.com/cloud/AmenesikServiceDashboard.pdf>
- Amenesik Cloud Operator (ACO) Version 1.1:  
<http://www.amenesik.com/cloud/AmenesikCloudOperator.pdf>

- Amenesik Platform Editor (APE) Version 1.1:  
<http://www.amenesik.com/cloud/AmenesikPlatformEditor.pdf>
- Amenesik Platform Service (APS) Version 1.1:  
<http://www.amenesik.com/cloud/AmenesikPlatformService.pdf>