



# DYNAMIC CONTEXT SERVER

4/24/2013

## C2M2L Final Integration Report

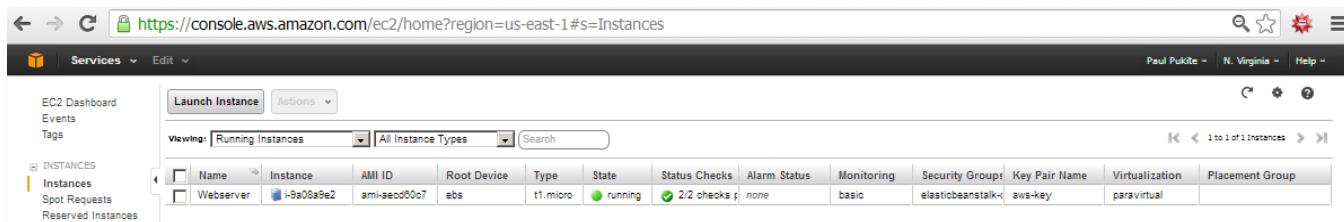
The Dynamic Context Server (DCS) is a semantic web application that provides ontologically-organized environmental modeling artifacts to users. The DCS can be hosted on a local personal computer or via a remotely served location. This report describes the steps needed to integrate the DCS within an Amazon Cloud environment.

# Dynamic Context Server

INTEGRATING THE DCS ON THE CLOUD

The development version of the DCS resides on either a Windows or Linux personal computer. To make it accessible to either larger or more geographically dispersed design teams, the plan was to eventually host it on a site such as Vehicle Forge (<http://VehicleForge.net>). In practical terms, Vehicle Forge duplicates the available cloud-hosted platform such as Amazon Web Services (<http://aws.amazon.com/>) allowing us to generalize the installation to an all-purpose hosting platform. As we see no loss of capability, for this integration report we describe the steps needed to import and deploy the DCS on a typical AWS node.

The first step is to obtain a web services account, select an EC2 operating system image, and start an instance according to the most recent instructions. Note the IP address selected for you. Interactive tools such as the EC2 Management Console are available to check the health of the instance session (see **Figure 1**)



**Figure 1: EC2 Management Console showing the named session**

Once started, open a terminal session on your local computer, and login to the EC2 instance by using **ssh** or its equivalent:

```
local> ssh user_name@host_address
```

The code for the DCS is available from either the delivered compressed archive, or from a subversion repository located here

[https://babelfish.arc.nasa.gov/trac/avm\\_performers/browser/context/](https://babelfish.arc.nasa.gov/trac/avm_performers/browser/context/)

Preliminary to the installation, unpack the archives or checkout the source from the subversion account.

Prerequisites for the installation are described in Annex 3 (titled “Installation”) of Appendix D of the C2M2L final report. This includes a short list of necessary open source supporting packages (swipl, R, graphviz).

These support applications needed by DCS may or may not be available as part of the ECS instance. If something is not available, we use the **yum** package management software to install the requisite apps.

To execute the DCS application, we have a choice to run it as a background daemon or as a no-hang-up (**nohup**) job, which will remain operational when we log out of the session. Since the latter is very convenient, we have provided an example **nohup** script called **nohup-run-cloud**. If launched from the main context directory, this will handle all the startup parameters.

```

pp@PP-HP ~
$ ssh -v ec2-user@23.23.137.157
OpenSSH_4.6p1, OpenSSL 0.9.8e 23 Feb 2007
debug1: Connecting to 23.23.137.157 [23.23.137.157] port 22.
debug1: Connection established.
debug1: identity file /c/Users/pp/.ssh/identity type -1
debug1: identity file /c/Users/pp/.ssh/id_rsa type 1
debug1: identity file /c/Users/pp/.ssh/id_dsa type -1
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.3
debug1: match: OpenSSH_5.3 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_4.6
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-cbc hmac-md5 none
debug1: kex: client->server aes128-cbc hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host '23.23.137.157' is known and matches the RSA host key.
debug1: Found key in /c/Users/pp/.ssh/known_hosts:3
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEKEYS sent
debug1: expecting SSH2_MSG_NEKEYS
debug1: SSH2_MSG_NEKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey
debug1: Next authentication method: publickey
debug1: Trying private key: /c/Users/pp/.ssh/identity
debug1: Offering public key: /c/Users/pp/.ssh/id_rsa
debug1: Server accepts key: pkalg ssh-rsa blen 277
debug1: read PEM private key done: type RSA
debug1: Authentication succeeded (publickey).
debug1: channel 0: new [client-session]
debug1: Entering interactive session.
Last login: Thu Apr 25 12:16:49 2013 from 63-224-20-49.mpls.qwest.net
[ec2-user@domU-12-31-39-02-76-AC ~]$
```

Amazon Linux AMI

<https://aws.amazon.com/amazon-linux-ami/2012.03-release-notes/>  
There are 53 security update(s) out of 332 total update(s) available  
Run "sudo yum update" to apply all updates.  
Amazon Linux version 2013.03 is available.

**Figure 2: Logging into an AWS session using a terminal emulator and SSH.**

```

[ec2-user@domU-12-31-39-02-76-AC ~]$ cd context/
[ec2-user@domU-12-31-39-02-76-AC context]$ pwd
/home/ec2-user/context
[ec2-user@domU-12-31-39-02-76-AC context]$ cat dynamic_context_server/config-enabled/network.pl
:- module(conf_network, []).
:- use_module(library(settings)).

/** <module> Configure the HTTP server
Change the default port on which the HTTP server listens. If
host-detection does not work or this server is behind a proxy, you may
also need the public_host/public_port settings.

The =prefix= setting rebases all paths on the server to the indicated
path. Note that the prefix has *no* trailing /. E.g. a setting =!/demo|=
changes the root of the server to =!/demo/. Rebasin a server is only
possible if internal path dependencies use the HTTP path mechanism to
find paths for internal services.

The setting =workers= sets the number of HTTP worker threads. See the
link below for more info.

@see   localhost.pl
@see   http_location_by_id/2 and http_link_to_id/3 for finding the
      locations of internal services.
@see   http://www.swi-prolog.org/howto/http/HTTPScale.html for more
      info on server scalability.
*/
:- set_setting_default(http:port, 80).
:- set_setting_default(http:public_host, 'entropplet.com').
:- set_setting_default(http:public_port, 80).
% :- set_setting_default(http:port, 8080).
% :- set_setting_default(http:public_host, 'www.example.org').
% :- set_setting_default(http:public_port, 80).
% :- set_setting_default(http:prefix, '/demo').
% :- set_setting_default(http:workers, 16).
[ec2-user@domU-12-31-39-02-76-AC context]$
```

**Figure 3 : The file network.pl is modified to use port 80 and a specific domain name.**

The script is launched from the `dynamic_context_server` subdirectory (see **Figure 4**). As a convenience, we execute as root to get access to port 80. Other methods are available to acquire that port without assuming root privileges, which is recommended to increase security.

```
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ ps -elf | grep swipl
4 S root      9067  8996  0  80  0 - 36999 poll_s 12:32 pts/1    00:00:00 sudo
nohup env PATH=/bin:/usr/local/bin:/usr/bin swipl -g cp_server,sleep(100000000)
-s run.pl -t halt
4 S root      9068  9067 28  80  0 - 236508 hrtime 12:32 pts/1    00:01:12 swipl
-g cp_server,sleep(100000000) -s run.pl -t halt
0 S ec2-user   9095  8996  0  80  0 - 25861 pipe_w 12:36 pts/1    00:00:00 grep
swipl
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ sudo kill -9 9068
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ cat nohup-run-cloud
sudo nohup env PATH=/bin:/usr/local/bin:/usr/bin swipl -g 'cp_server,sleep(1000
0000)' -s run.pl -t halt &
[1]+  Exit 137                  sudo nohup env PATH=/bin:/usr/local/bin:/usr/bin
swipl -g 'cp_server,sleep(100000000)' -s run.pl -t halt
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ ./nohup-run-cloud
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ nohup: ignoring input
and appending output to 'nohup.out'
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$
```

**Figure 4 : To run a DCS application, we first terminate any existing apps identified by “swipl” and then launch the “nohup-run-cloud” script. The script is essentially a one-liner which will run the application in the background and save output to a log file.**

To terminate a session, use the `kill` command on the launched process id (PID). Use the commands `ps` or `top` to identify the PID of the server. There are no mechanisms for restarting a DCS session, other than to terminate and relaunch via the script. Monitoring of the application output is available through tailing the `nohup.out` log file (see **Figure 5**).

```
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$ sudo tail -9 nohup.out
% /home/ec2-user/context/dynamic_context_server/run.pl compiled 2.56 sec, 31,375
clauses
% Started ClioPatria server at port 80
% You may access the server at http://entropplet.com:80/
% Restoring 253 snapshots using 1 concurrent workers
combined_ontology.rdf..... 253 of 253 graphs
% Restoring 1 journals using 1 concurrent workers
user..... 1 of 1 graphs
% Loaded 254 graphs (436,253 triples) in 6.89 sec. (44% CPU = 3.02 sec.)
[ec2-user@domU-12-31-39-02-76-AC dynamic_context_server]$
```

**Figure 5 : The output of the log file can be monitored via “tail -f”. Otherwise, it is safe to log out from the EC2 session at this point and the app will continue to execute.**

To verify a successful integration, navigate to the IP address of the AWS EC2 instance using a Firefox, Chrome, or recent Internet Explorer browser (see **Figure 6**).

**Dynamic Context Server - Features**

- [Requirements](#) - ▾
- [Search](#) - ▾
- [Browse](#) - ▾
- [Workflows](#) - ▾
- [References](#) - ▾
- [Resources / Artifacts](#) - ▾
- [Map View](#) - ▾
- [Generic Query](#) - ▾
- [Features Home](#) - ▾

**Environmental Context Server Features**

The dynamic context server provides interactive content for environmental modeling. The models are contained within an ontologically organized knowledgebase and so can be semantically accessed. The semantic organization allows various search mechanisms while the modeling domains are roughly grouped according to the icons described below. A context is defined as the surrounding environment for a specific intended use, such as for evaluating vehicle design or performance.

[Stochastic Modeling | Ontological Organization and Process Modeling](#)

 [Fine Terrain Features](#) - Models for representing power spectral densities (PSD) based on semi-Markov approaches. Models of obstacle courses and data sets for friction and soil types.

 [Gross Terrain Features](#) - Simple distribution for sampling likelihood of terrain slopes. Marginal distributions for elevation changes over various geographic locations.

 [Wave Energy Statistics](#) - Simple approach for modeling PDFs of wave frequencies and wave heights. Models of sea-state wave distributions over various geographic locations. Water density and buoyancy tables.

 [Wind Energy Statistics](#) - General results for modeling wind energy distribution and autocorrelations for temporal persistence. Solar insolation models.

 [EMI Clutter Modeling](#) - Maximum entropy models for E-M signals and noise. Models of lightning rate.

 [Inland-water Statistics](#) - Maximum entropy estimation for lake size distributions and river flow rates over various regions.

 [Particle Size Statistics](#) - General method for modeling size distribution of particulates such as ash and ice crystals, and water droplets.

dynamic context server

**Figure 6: Home screen of the Dynamic Context server**

As delivered, the default DCS administrative user name is **pp**, and password **pukite**. This is required to load the repository context data, via the menu command shown in **Figure 7**.

**Dynamic Context Server**

- [Requirements](#) - ▾
- [Search](#) - ▾

**Repository**

- Load Context Data
- Load All Ontology
- Zotero Ref Library
- Load local file
- Load from HTTP
- Load from library
- Remove triples
- Clear repository

**Environmental Context Server**

The dynamic context server provides interactive content for environmental modeling. The models are contained within an ontologically organized knowledgebase and so can be semantically accessed. The semantic organization allows various search mechanisms while the modeling domains are roughly grouped according to the icons described below. A context is defined as the surrounding environment for a specific intended use, such as for evaluating vehicle design or performance.

**Figure 7 : As a one-time step, the semantic context data needs to be loaded administratively. This will normally take a few seconds at most. Regular users can not access the repository functionality.**

The EC2 platform features support such as fail-over whereby a session will automatically restart should it terminate unexpectedly.

That concludes a description of the steps needed to integrate the Dynamic Context Server onto a cloud-based Amazon EC2 platform. Users can then access the DCS by navigating to the chosen IP.