# FICO® Xpress Optimization

Last update February 2018

2.0

REFERENCE MANUAL

## AEC2: Managing Amazon EC2 from Mosel

FICO Decisions

AEC2

Deliverable Version: A

Last Revised: February 2018

Version 2.0

# Contents

CHAPTER 1

# Introduction

Amazon Elastic Compute Cloud (EC2) is part of Amazon Web Services (AWS http://aws.amazon.com). Using either a web interface or a set of API calls the service allows to run virtual machines on the Amazon infrastructures. A virtual machine *instance* is started from an Amazon Machine Image (AMI): this is the image of the machine to run (it includes both the operating system and a set of pre-installed applications).

The AEC2 Mosel package is a subset of the AWS API for Mosel. It enables a model to:

- manage basic AWS functionality like creating keys, defining security credentials, building AMIs, ...
- start/stop and query the status of running virtual machines in the Cloud
- start and use Mosel instances on virtual machines just like they were running on the local network (using the standard *mmjobs* functionality)

This package is also available as an extension to XPRD (Java version) with which a Java program using XPRD can manage instances created in EC2. This extension uses the same configuration files as the Mosel package.

## 1.1 Overview

The AEC2 package consists in a Mosel package (`aec2.bim`) and the program `aec2setup.mos`. In addition to these components, system commands to connect to remote hosts using the SSH protocol are also required. On Unix-like platforms, the programs `ssh` and `scp` are usually available; on Windows systems, specific programs `pcmdgen.exe`, `mpscp.exe` and `mplink.exe` are included in the package distribution.

Using Amazon EC2 requires the availability of machine images (AMI) in order to start virtual machines. For the particular case of Mosel, the AMI must have Xpress installed and accept connections for running remote Mosel instances either via SSH or using the *xprmsrv* server: the provided setup program (`aec2setup.mos`) can be used to create appropriate images. This program, that is to be run once only, also results in a set of configuration files required to initialise the package when it is used by a model.

With the help of the *mmjobs* module a Mosel model can already start and manage remote instances for creating distributed applications. The AEC2 package adds a set of routines to this framework for managing EC2 virtual machines and generating the appropriate connection strings for the `connect` function of *mmjobs* AEC2 is designed such that enabling an existing distributed model to use resources in the Cloud merely requires the user to add a few statements to the original Mosel model source code. For instance, consider the following model that compiles and then executes the model `"mymodel.mos"` on some remote host `"myserver"`:

```
model dist
uses 'mmjobs'

declarations
 minst: Mosel
 rmod: Model
end-declarations

if connect(minst, "myserver")<>0 then
 exit(1)
end-if

if compile(minst, "", "rmt:mymodel.mos", "tmp:m.bim")<>0 then
 exit(2)
end-if

load(minst, rmod, "tmp:m.bim")
run(rmod)
wait
dropnextevent
unload(rmod)
disconnect(minst)
end-model
```

The following example performs the same operations but using a virtual machine in EC2 instead of a local server:

```
model distec2
uses 'mmjobs','aec2'

declarations
 ainst: EC2instance
 minst: Mosel
 rmod: Model
end-declarations

if not loadAEC2Config("aec2.acf") then  ! load AEC2 configuration
 exit(-1)
end-if

ainst:=runInstance                      ! start a virtual machine in the Cloud
if not waitInstanceReady(ainst, 300, true) then
 exit(-2)
end-if

if connect(minst,getConnString(ainst))<>0 then
 exit(1)
end-if

if compile(minst, "", "rmt:mymodel.mos", "tmp:m.bim")<>0 then
 exit(2)
end-if

load(minst, rmod, "tmp:m.bim")
run(rmod)
wait
dropnextevent
unload(rmod)
disconnect(minst)

terminateInstance(ainst)                ! terminate the virtual machine
end-model
```

Note the use of function `getConnString` as input parameter for the *mmjobs* function `connect`: it generates a valid connection string for the specified Amazon instance (either for an SSH or *xprmsrv* link depending on the default protocol).

## 1.2 Setup and creation of an Amazon Machine Image (AMI) for Mosel

The AEC2 settings are stored in a set of configuration files. The `'aec2setup.mos'` program will configure your AWS account as needed and create these configuration files. One of the tasks performed by this program is to create a dedicate AMI running the Linux operating system. To this end, an Xpress Linux package (either 32bit or 64bit depending on the needs) together with its associated license file are required to run the model.

To run the setup program, copy into the same directory the Xpress Linux archive (file name in the form `"xp?.?.?_linux*_setup.tar"`), the `"xpauth.xpr"` license and the `"aec2setup.mos"` program itself then, from a command prompt type:

```
mosel -s -c "exe aec2setup"
```

During the process, the program will ask various questions, in most cases a default answer is provided (noted in square brackets): simply hit enter if you do not know what to do. As part of the procedure a virtual machine is started in the Cloud and Xpress is installed on it in order to create the AMI to be used later on. Note that:

- starting a virtual machine and creating the AMI may be quite long operations (up to several minutes): do not interrupt the program during these operations
- in case of failure, you can restart the program: it will recover information already setup.

Once the setup has completed, the following files have been created:

- `"aec2.aid"` the secret key-pair to access your AWS account. This file must be kept secret as it gives full access to your AWS account
- `"aec2.ask"` the private key to log into virtual machines. The information stored in this file cannot be regenerated. If it is lost or corrupted, a new security credential will have to be created (*i.e.* delete the file and re-run `aec2setup`)
- `"aec2.acf"` the main configuration file (that references the 2 others)

These 3 files are required by the AEC2 package when running a model using it.

CHAPTER 2

# Main functions of the AEC2 package

## 2.1 Public declarations

An AMI is represented by the `EC2image` record type described as follows:

```
id:string
name:text
description:text    ! "Xpress for AEC2" for images created by aec2setup
location:text
snapshot:text
available:boolean
ispublic:boolean
architecture:string
owner:string
ownerid:string
```

A running (or recently terminated) instance is represented by the `EC2instance` record type publishing the following fields:

```
image:string        ! image name
state:integer       ! AEC2_PENDING/AEC2_RUNNING/AEC2_SHUTTING_DOWN/
                    ! AEC2_STOPPED/AEC2_STOPPING/AEC2_TERMINATED
dns:text            ! public DNS name
key:text            ! key used for ssh connections
type:string         ! e.g. "t1.micro"
launchtime:datetime
ipaddr:text         ! public IP
```

The package also publishes the following global variables:

```
aec2_image:string     ! image name
aec2_type:string      ! instance type (e.g. "t1.micro")
aec2_secgrp:string    ! security group for SSH sessions
aec2_connmode:string  ! connecion mode: "xprmsrv" or "ssh"
aec2_xsrvctx:string   ! xprmsrv context
aec2_xsrvpass:string  ! xprmsrv password
aec2_xsrvport:integer ! xprmsrv connection port
```

These variables are initialised when loading the configuration file using `loadAEC2Config` but may be modified directly after this routine has been called.

## 2.2 Procedures and functions

# getAllInstances

### Purpose

Retrieve a list of Instances from AWS.

### Synopsis

```
function getAllInstances:list of EC2instance
```

### Return value

List of running (or recently terminated) instances.

### Further information

The returned list may contain instances that have already terminated. It is important to check the actual state of an instance before using it (field `state` of the `EC2instance` type).

### Related topics

`getImages`.

## getConnString

### Purpose

Generate a connection string.

### Synopsis

```
function getConnString(inst:EC2instance):string
```

### Argument

`inst` Target instance

### Return value

A connection string suitable for the `connect` routine of *mmjobs*.

### Further information

This function calls either `getConnStringSSH` or `getConnStringXSRV` depending on the value of the variable `aec2_connmode`.

### Related topics

`getConnStringXSRV`, `getConnStringSSH`.

## getConnStringSSH

### Purpose

Generate a connection string for an SSH link.

### Synopsis

```
function getConnStringSSH(inst:EC2instance):string
```

### Argument

`inst` Target instance

### Return value

A connection string suitable for the `connect` routine of *mmjobs*.

### Further information

The resulting connection string is of the form `"rcmd:mplink -batch -ssh -i keyfile user@ipaddr mosel -r"` on Windows systems and `"rcmd:ssh -o StrictHostKeyChecking=no -i keyfile user@ipaddr mosel -r"` on Unix-like operating systems.

### Related topics

`getConnStringXSRV`, `getConnString`.

# getConnStringXSRV

### Purpose

Generate a connection string for an *xprmsrv* server.

### Synopsis

```
function getConnStringXSRV(inst:EC2instance):string
```

### Argument

`inst` Target instance

### Return value

A connection string suitable for the `connect` routine of *mmjobs*.

### Further information

The resulting connection string is of the form `"xsrv:ipaddr(port)/ctx/pass"`.

### Related topics

`getConnStringSSH`, `getConnString`.

## getImages

### Purpose

Retrieve a list of AMI from AWS.

### Synopsis

```
function getImages(Id:string, name:string, desc:string):list of EC2image
```

### Arguments

| | |
|---|---|
| `Id` | Image ID (or an empty string) |
| `name` | Image name (or an empty string) |
| `desc` | Image description (or an empty string) |

### Return value

List of images with the specified properties.

### Example

The following example retrieves all images created by `aec2setup` and the specification of the default image used to start new instances:

```
xpimgs:=getImages("","","Xpress for AEC2*")
defimg:=getImages(aec2_image,"","")
```

### Further information

1. Each of the parameters is used as a filter: the resulting list will keep only images with the corresponding property matching the filter. Filters support wildcard charcaters `"*"` (any sequence of character possibly empty) and `"?"` (any character). An empty string is equivalent to `"*"` (any string).
2. Images created by the program `aec2setup` have the string `"Xpress for AEC2"` included in their description: this can be used to find existing images compatible with the package.

### Related topics

`getAllInstances`.

## loadAEC2Config

### Purpose

Load a configuration file and initialise the package.

### Synopsis

```
function loadAEC2Config(confFile:string):boolean
```

### Argument

`confFile` Configuration file name

### Return value

`true` if operation succeeded.

### Further information

This function must be called before any other routine of the package. The configuration files are created by running the `"aec2setup.mos"` program.

## runInstance

### Purpose

Start a new instance in EC2.

### Synopsis

```
function runInstance:EC2instance
```

### Return value

Instance record referencing the created instance.

### Further information

This function starts a new instance using the current configuration (in particular using 'aec2_image' as the image). Note that an instance needs some time before it is ready to execute commands — this routine does not wait for the instance to be ready: use `waitInstanceReady` for this purpose.

### Related topics

`waitInstanceReady`, `updateInstance`.

## terminateInstance

### Purpose

Terminate the specified instance.

### Synopsis

```
procedure terminateInstance(inst:EC2instance)
```

### Argument

`inst` Instance to terminate

### Related topics

`runInstance`.

## updateInstance

### Purpose

Update instance information.

### Synopsis

```
procedure updateInstance(inst:EC2instance)
```

### Argument

`inst` Instance object to update

### Further information

An `EC2instance` object contains information on an instance running in EC2. This information is not necessarily up to date: this procedure queries AWS and updates properties stored in the corresponding record.

## waitInstanceReady

### Purpose

Wait for an instance to be ready.

### Synopsis

```
function
      waitInstanceReady(inst:EC2instance,delay:integer,verbose:boolean):boolean
```

### Arguments

| | |
|---|---|
| `inst` | Instance to wait for |
| `delay` | Maximum amount of time (seconds) to wait |
| `verbose` | Display progress bar if `true` |

### Return value

`true` if the instance was ready before the expiration of the specified delay.

### Further information

This function waits at most `delay` seconds for the instance `inst` to be ready. If `verbose` is `true`, a dot will be displayed at regular intervals.

### Related topics

`runInstance`, `updateInstance`.

## Appendix A

# Contacting FICO

FICO provides clients with support and services for all our products. Refer to the following sections for more information.

## Product support

FICO offers technical support and services ranging from self-help tools to direct assistance with a FICO technical support engineer. Support is available to all clients who have purchased a FICO product and have an active support or maintenance contract. You can find support contact information and a link to the Customer Self Service Portal (online support) on the Product Support home page (www.fico.com/en/product-support).

The FICO Customer Self Service Portal is a secure web portal that is available 24 hours a day, 7 days a week from the Product Support home page. The portal allows you to open, review, update, and close cases, as well as find solutions to common problems in the FICO Knowledge Base.

Please include *'Xpress'* in the subject line of your support queries.

## Product education

FICO Product Education is the principal provider of product training for our clients and partners. Product Education offers instructor-led classroom courses, web-based training, seminars, and training tools for both new user enablement and ongoing performance support. For additional information, visit the Product Education homepage at www.fico.com/en/product-training or email producteducation@fico.com.

## Product documentation

FICO continually looks for new ways to improve and enhance the value of the products and services we provide. If you have comments or suggestions regarding how we can improve this documentation, let us know by sending your suggestions to techpubs@fico.com.

Please include your contact information (name, company, email address, and optionally, your phone number) so we may reach you if we have questions.

## Sales and maintenance

If you need information on other Xpress Optimization products, or you need to discuss maintenance contracts or other sales-related items, contact FICO by:

- Phone: +1 (408) 535-1500 or +44 207 940 8718
- Web: http://www.fico.com/optimization and use the available contact forms

## Related services

**Strategy Consulting:** Included in your contract with FICO may be a specified amount of consulting time to assist you in using FICO Optimization Modeler to meet your business needs. Additional consulting time can be arranged by contract.

**Conferences and Seminars:** FICO offers conferences and seminars on our products and services. For announcements concerning these events, go to www.fico.com or contact your FICO account representative.

## FICO Community

The FICO Community is a great resource to find the experts and information you need to collaborate, support your business, and solve common business challenges. You can get informal technical support, build relationships with local and remote professionals, and improve your business practices. For additional information, visit the FICO Community (community.fico.com/welcome).

## About FICO

FICO (NYSE:FICO) powers decisions that help people and businesses around the world prosper. Founded in 1956 and based in Silicon Valley, the company is a pioneer in the use of predictive analytics and data science to improve operational decisions. FICO holds more than 165 US and foreign patents on technologies that increase profitability, customer satisfaction, and growth for businesses in financial services, telecommunications, health care, retail, and many other industries. Using FICO solutions, businesses in more than 100 countries do everything from protecting 2.6 billion payment cards from fraud, to helping people get credit, to ensuring that millions of airplanes and rental cars are in the right place at the right time. Learn more at www.fico.com.

# Index