vmcaster

2014-01-21

Virtualisation Working Group Format HEPiX

Owen Synge

vmcaster: 2014-01-21

Virtualisation Working Group Format HEPiX
Owen Synge

Abstract

Virtual Maschine Image list Format as developed by the HVG as an on going document.

Table of Contents

Preface

Table of Contents

`Virtual Machine Image List` are a extendible meta data format used to transfer `Virtual Machine Image` from end user to cloud resource provider. `Virtual Machine Image List` within a working group of HEPiX. HEPiX is a twice yearly conference made up from sites providing compute and storage services for High Energy Physics. HEPiX focusses on the compute, network and storage infrastucture required for High Energy Physics sites.

HEPiX Virtualisation Working Group Observations

The spring 2009 HEPiX conference made the following observations:

- High Energy Physics will want consistent images across sites.
- The Grid is a homogeneous OS environment so requiring High Energy Physics to coordinate OS upgrades across multiple experiments.
- The Grid will need to be shared with non High Energy Physics community.
- High Energy Physics experiments are very coupled to OS.
- Hardware is coupled to OS version.
- Sites believe `Virtual Machine Image`'s are going to be the execution environment for High Energy Physics jobs in the future.
- Change Root, and Xen based `Virtualised Execution Environments` where already present at HEPiX sites.

With these observations the HEPiX Virtualisation Working Group was formed.

A new generation of `Virtualised Execution Environments` such as OpenStack, Open Nebula and Nimbus where already being developed.

HEPiX Virtualisation Working Group Objectives

- Produce a framework to securely run `Virtual Machine Image`'s across multiple sites supporting High Energy Physics.
- Sites need control over `Virtual Machine Image` selection.

With these objectives the HEPiX Virtualisation Working Group was created with Tony Cass as its leader.

Chapter 1. Virtual Machine Image List Subscribing Software

Owen Synge

vmcatcher

This virtual machine `Virtual Machine Image List` subscriber implementation is intended to be a production grade reference implementation.

The software makes use of a database to store subscriptions in a similar way to a podcast reader or a Linux package manager. The tested Database is SqlLight, but it is based upon SQLalchamy so should support many databases. Sdllight has proved more than adequate for the low transaction rate of a image list subscriber and so deployment issues are just backing up a databasefile.

Since the software is made with the Grid in mind it is only natural that the `x.509` certificate model is used.

Introduction

This application allows users to subscribe to `Virtual Machine Image List`s, cache the images referenced to in the `Virtual Machine Image List`, validate the images list with `x.509` based public key cryptography, and validate the images against sha512 hashes in the images lists and provide events for further applications to process update or expire changes of virtual machine images without having to further validate the images.

This software is available at:
`https://github.com/hepix-virtualisation/vmcatcher`

The software is based upon a simple database that stores subscriptions to `Virtual Machine Image List`s, who can sign the `Virtual Machine Image List`, and which images belong to which subscriptions. It allows images to selected for subscription.

Subscribed images can be downloaded verified and cached. Cached images can be verified, and if invalid or expired they are moved to an expiry directory.

Features

- Add and delete multiple subscriptions to `Virtual Machine Image List`s.
- Update subscriptions checking authenticity of the message using `x.509` based signatures.
- Automation as a cron script.
- Subscribe and unsubscribe to images from `Virtual Machine Image List`s.
- Download verify images into a local cache.
- Expire images to an archive when no longer endorsed or corrupt.
- Open Stack intgration by Mattieu Puel CC-IN2P3.fr.

- OpenNebula intgration by Roberto Rosende Dopazo CESGA.es.

This set of applications are designed to provide a similar work flow from each area of control to the `Virtual Machine Image List` archive.

- `vmcatcher_endorser` - Endorsers of `Virtual Machine Image List` subscriptions.
- `vmcatcher_subscribe` - Subscription list details.
- `vmcatcher_image` - Image details.
- `vmcatcher_cache` - Cache images and update events.

They work in conjunction with a database to ease navigation, a local cache of `Virtual Machine Image List` subscriptions. The database is message format agnostic, but it authenticates and validates all messages in import. Because these are just caches of `Virtual Machine Image List`s they are meant to be used the majority of the time without intervention.

If you are signing a list using the HEPiX `Virtual Machine Image List` signer, you should also install this application and subscribe to your current image.

It is intended to with a couple of cron scripts to be informed at any time if your local images are matching signatures in the `Virtual Machine Image List`.

Anyone curious about this application should consider this application a software application similar to Debian's 'aptitude' or Redhats 'yum', but rather for virtual machines, authenticated by the `x.509` signatures.

Like yum and apt this tool can be extended. The following handlers have been developed

Table 1.1. Cloud Integration with vmcatcher

Cloud	Name	Author	Employer	URI
Open Stack	glancepush-vm-catcher	Mattieu Puel	CC-IN2P3.fr	https://github.com/EGI-FCTF/glance-push-vmcatcher
Open Stack	glancepush	Mattieu Puel	CC-IN2P3.fr	https://github.com/EGI-FCTF/glance-push
OpenNebula	Cesga cloud tools	Roberto Rosende Dopazo	Cesga.es	https://github.com/grid-admin/cloud

Quick start use of vmcatcher

First make sure that all the Certificate Revocation Lists (CRL) are upto date.

```
[root] #  fetch-crl
```

This suit of applications can use either environment variable or command line to set most parameters. If neither environment variables or command line parameters are not set for critical variables the application will provide defaults and show warnings.

The most important setting is the location of the database. This is read from VMCATCHER_RDBMS,

```
[user] $ export VMCATCHER_RDBMS="sqlite:////var/lib/vmcatcher/vmcatcher.db"
```

The above line instructs the SQLalchamy interface to databases to use sqlite and path "/var/lib/vmcatcher/vmcatcher.db" on a UNIX system. This is the only important file and stores the older signed image lists. SQL is used to enforce most of the rules such as unique nature of `RFC 4122 UUID`'s and the URLS for the subscriptions, this should be backed up. Other environment variables are documented later.

To add a subscription,

```
[user] $  wget --no-check-certificate https://cernvm.cern.ch/releases/image.list
```

Now you can check the `Virtual Machine Image List` by visual inspection.

```
[user] $  grep 'hv:[cd][an]' hepix_signed_image_list
                "hv:ca": "/DC=ch/DC=cern/CN=CERN Trusted Certification Authority",
                "hv:dn": "/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=buncic/CN=379010/CN=Predrag
 Buncic",
```

Now create this endorser. The endorser_uuid can be any string but its recommended this is a short string possibly following the `RFC 4122 UUID` standard:

```
[user] $  vmcatcher_endorser --create \
       --endorser_uuid='Predrag Buncic' \
       --subject='/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=buncic/CN=379010/CN=Predrag Buncic'
 \
       --issuer='/DC=ch/DC=cern/CN=CERN Trusted Certification Authority'
```

Now we can add the subscription, this will automatically link the endorser with this subscription.

```
[user] $  vmcatcher_endorser -l
Ian    '/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=buncic/CN=379010/CN=Predrag Buncic'        '/
DC=ch/DC=cern/CN=CERN Trusted Certification Authority'
```

The above command will show you the endorsers. Note the first column is the identifier. In this case its shorter than a `RFC 4122 UUID`. The second column is the users certificate subject (some times know as distinguished name) while the third column is the subject of the issuing certificate authority.

```
[user] $  vmcatcher_subscribe -s file:////`pwd`/hepix_signed_image_list
INFO:main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
WARNING:db_actions:list hv:uri does not match subscription uri
```

Although less secure it is also possible to add the option '--auto-endorse' to the command line so that both the endorsers, and endorsers issuer's certificate subjects are added to the database automatically. This is particularly useful for testing.

```
[user] $  vmcatcher_subscribe  --auto-endorse -s file:////`pwd`/hepix_signed_image_list
INFO:main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
WARNING:db_actions:list hv:uri does not match subscription uri
```

List the registered Images.

```
[user] $  vmcatcher_image -l
INFO:vmcatcher_subscribe.main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
327016b0-6508-41d2-bce0-c1724cb3d3e2    0       63175437-7d59-4851-b333-c96cb6545a86
858a817e-0ca2-473f-89d3-d5bdfc51968e    0       63175437-7d59-4851-b333-c96cb6545a86
da42ca85-179b-4873-b12e-32d549bf02b6    0       63175437-7d59-4851-b333-c96cb6545a86
```

The results show the RFC 4122 UUID of the image, the availability state and the subscription RFC 4122 UUID. The state value is a bitmap;

- 1 Image is subscribed

- 2 Image is available from a valid Virtual Machine Image Lists.

Now we will select an image for local caching.

Next update the subscriptions.

```
[user] $  vmcatcher_subscribe -U
INFO:main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
INFO:db_actions:Updating:63175437-7d59-4851-b333-c96cb6545a86
```

Now the data base contains the latest version of the Virtual Machine Image List. To list the available images referenced in the local database:

```
[user] $  vmcatcher_image -l
INFO:vmcatcher_subscribe.main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
327016b0-6508-41d2-bce0-c1724cb3d3e2    2       63175437-7d59-4851-b333-c96cb6545a86
858a817e-0ca2-473f-89d3-d5bdfc51968e    2       63175437-7d59-4851-b333-c96cb6545a86
da42ca85-179b-4873-b12e-32d549bf02b6    2       63175437-7d59-4851-b333-c96cb6545a86
```

This now shows the images are available in the latest Virtual Machine Image List.

```
[user] $  vmcatcher_image -a -u 858a817e-0ca2-473f-89d3-d5bdfc51968e
INFO:vmcatcher_subscribe.main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
```

The Virtual Machine Image List state is now changed to

```
[user] $  vmcatcher_image -l
INFO:vmcatcher_subscribe.main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
327016b0-6508-41d2-bce0-c1724cb3d3e2    2       63175437-7d59-4851-b333-c96cb6545a86
858a817e-0ca2-473f-89d3-d5bdfc51968e    3       63175437-7d59-4851-b333-c96cb6545a86
da42ca85-179b-4873-b12e-32d549bf02b6    2       63175437-7d59-4851-b333-c96cb6545a86
```

Clearly showing that the image ’858a817e-0ca2-473f-89d3-d5bdfc51968e’ is subscribed.

Make the directories for caching the images.

```
[user] $  mkdir cache cache/partial cache/expired
```

Now cache the images.

```
[user] $  vmcatcher_cache
INFO:vmcatcher_subscribe.main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
INFO:DownloadDir:Downloading '858a817e-0ca2-473f-89d3-d5bdfc51968e'.
INFO:CacheMan:moved file 858a817e-0ca2-473f-89d3-d5bdfc51968e
```

Once this is complete the image from the Virtual Machine Image List will be cached.

```
[user] $  find cache/
cache/
cache/partial
cache/partial/cache.index
cache/expired
cache/expired/cache.index
```

```
cache/858a817e-0ca2-473f-89d3-d5bdfc51968e
cache/cache.index
```

Installation

The latest build system artefacts are published here http://www.yokel.org/pub/software/yokel.org/release/
.

Package Repositories.

The intra site tools are tested on every release for Redhat Enterprise Linux 6 and are developed on the Debian Linux platform. They are available as src and binary RPM packages in the following repository sporting.

- http://www.yokel.org/pub/software/yokel.org/release/x86_64/scientific/6x/rpm/
- http://grid.desy.de/vm/repo/yum/sl5/noarch/RPMS.stable/

Prebuilt Debian packages is currently work in progress, but prebuilt tar balls are available.

Deployment instructions are provided in the README included in the source code and the RPM.

Installation on Redhat Enterprise Linux 6

Install EPEL for dependencies.

```
[root] # rpm -i http://download.fedora.redhat.com/pub/epel/6/x86_64/epel-release-6-5.noarch.rpm
```

Install Yokel yum repository.

```
[root] # cat /etc/yum.repos.d/vmcasting.repo
[vmcasting]
name=vmcasting
baseurl=http://www.yokel.org/pub/software/yokel.org/release/x86_64/scientific/6x/rpm/
enabled=1
gpgcheck=0
```

Install the Grid CA repository for details please see https://wiki.egi.eu/wiki/EGI_IGTF_Release

```
[root] # cat /etc/yum.repos.d/egi-trust-anchor.repo
[EGI-trustanchors]
name=EGI-trustanchors
baseurl=http://repository.egi.eu/sw/production/cas/1/current/
gpgkey=http://repository.egi.eu/sw/production/cas/1/GPG-KEY-EUGridPMA-RPM-3
gpgcheck=1
enabled=1
```

install the ca-policy-egi-core

```
[root] # yum install ca-policy-egi-core
```

install fetch-crl

```
[root] # yum install fetch-crl
```

```
[root] # yum install vmcatcher
```

Installation on Redhat Enterprise Linux 5

Redhat Enterprise Linux 5 is no longer updated please report a bug if you still need this platform.

Install EPEL for dependencies.

```
[root] # rpm -i http://download.fedora.redhat.com/pub/epel/5/x86_64/epel-release-5-4.noarch.rpm
```

Install DESY yum repository.

```
[root] # cat /etc/yum.repos.d/vmcasting.repo
[vmcasting]
name=vmcasting
baseurl=http://grid.desy.de/vm/repo/yum/sl5/noarch/RPMS.stable/
enabled=1
gpgcheck=0
```

Install the Grid CA repository for details please see https://wiki.egi.eu/wiki/EGI_IGTF_Release

```
[root] # cat /etc/yum.repos.d/egi-trust-anchor.repo
[EGI-trustanchors]
name=EGI-trustanchors
baseurl=http://repository.egi.eu/sw/production/cas/1/current/
gpgkey=http://repository.egi.eu/sw/production/cas/1/GPG-KEY-EUGridPMA-RPM-3
gpgcheck=1
enabled=1
```

Install the lcg-CA

```
[root] # yum install lcg-CA
```

install fetch-crl

```
[root] # yum install fetch-crl
```

Install the HEPiX `Virtual Machine Image List` subscriber.

```
[root] # yum install vmcatcher
```

This may fail due to a dependency of m2crypto that cannot be satisfied. This is due to known bugs in m2crypto in the version shipped in RHEL5. If this is a problem please download the following

```
http://ftp.informatik.uni-frankfurt.de/fedora-archive/fedora/linux/releases/8/Everything/source/
SRPMS/m2crypto-0.18-2.src.rpm
```

And build a native RPM.

Installation on Debian Linux 7.0 (Wheezy) or later,

Do not install this on Debian 6.0 as the included version of python-m2crypto is not stable.

These instructions are for Debian Linux 7.0 (Wheezy) or later.

Unfortunately at this moment the code is not packaged, but they will be soon. All the dependencies are available in the Debian repository.

For Grid scientific use you can get a trust store easily using the egi.eu repository.

```
[root] # wget -q -O - \
https://dist.eugridpma.info/distribution/igtf/current/GPG-KEY-EUGridPMA-RPM-3 \
 | apt-key add -
```

Add the following line to your sources.list file for APT:

```
#### EGI Trust Anchor Distribution ####
deb http://repository.egi.eu/sw/production/cas/1/current egi-igtf core
```

for example:

```
[root] # echo '#### EGI Trust Anchor Distribution ####' >> \
    /etc/apt/sources.list
[root] # echo 'deb http://repository.egi.eu/sw/production/cas/1/current egi-igtf core' >> \
    /etc/apt/sources.list
```

Now install the `Certification Authorities` for the grid (Other `Certification Authorities` can be substituted), install a tool to download and cache the `Certificate Revocation Lists`

```
[root] # aptitude update
[root] # aptitude install ca-policy-egi-core
[root] # aptitude install fetch-crl
[root] # fetch-crl
```

Now install the code from git.

```
http://www.yokel.org/pub/software/yokel.org/release/source/debian/7.0/tgz/
```

The latest version of hepixvmitrust-X.X.XX.src.tar.gz should be downloaded extracted and installed.

```
[root] # wget http://www.yokel.org/pub/software/yokel.org/release/source/debian/7.0/tgz/
hepixvmitrust-0.0.15.src.tar.gz
Resolving grid.desy.de (grid.desy.de)... 131.169.180.46
Connecting to grid.desy.de (grid.desy.de)|131.169.180.46|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19922 (19K) [application/x-tar]
Saving to: `hepixvmitrust-0.0.15.src.tar.gz'

100%[======================================>] 19,922      --.-K/s   in 0.05s

2012-05-28 19:45:45 (413 KB/s) - `hepixvmitrust-0.0.15.src.tar.gz' saved [19922/19922]
[root] # tar -zxf hepixvmitrust-0.0.15.src.tar.gz
[root] # cd hepixvmitrust-0.0.15
[root] # python setup install
[root] # echo $?
[root] # cd ..
```

The latest version ofsmimeX509validation-0.0.11.src.tar.gz -X.X.XX.src.tar.gz should be downloaded extracted and installed.

```
[root] #
[root] # wget http://www.yokel.org/pub/software/yokel.org/release/source/debian/7.0/tgz/
smimeX509validation-0.0.11.src.tar.gz
Resolving grid.desy.de (grid.desy.de)... 131.169.180.46
Connecting to grid.desy.de (grid.desy.de)|131.169.180.46|:80... connected.
```

```
HTTP request sent, awaiting response... 200 OK
Length: 19922 (19K) [application/x-tar]
Saving to: `smimeX509validation-0.0.11.src.tar.gz'

100%[=====================================>] 19,922      --.-K/s   in 0.05s

2012-05-28 19:45:45 (413 KB/s) - `smimeX509validation-0.0.11.src.tar.gz' saved [19922/19922]
[root] # tar -zxf smimeX509validation-0.0.11.src.tar.gz
[root] # cd smimeX509validation-0.0.11
[root] # python setup install
[root] # echo $?
[root] # cd ..
```

The latest version of vmcatcher-X.X.XX.src.tar.gz should be downloaded extracted and installed.

```
[root] #
[root] # wget http://www.yokel.org/pub/software/yokel.org/release/source/debian/7.0/tgz/
vmcatcher-0.1.29.src.tar.gz
Resolving grid.desy.de (grid.desy.de)... 131.169.180.46
Connecting to grid.desy.de (grid.desy.de)|131.169.180.46|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19922 (19K) [application/x-tar]
Saving to: `vmcatcher-0.1.29.src.tar.gz'

100%[=====================================>] 19,922      --.-K/s   in 0.05s

2012-05-28 19:45:45 (413 KB/s) - `vmcatcher-0.1.29.src.tar.gz' saved [19922/19922]
[root] # tar -zxf vmcatcher-0.1.29.src.tar.gz
[root] # cd vmcatcher-0.1.29
[root] # python setup install
[root] # echo $?
[root] # cd ..
```

Environment Variables

Environment variables can be used to set default values but the command line options will override any set environment options.

VMCATCHER_RDBMS

Sets the path to the database. For example "sqlite:///vmcatcher.db"

VMCATCHER_CACHE_EVENT

Sets the executions string. Command line options can be set as environment variables just like the command line interface. Users of the "sh shell" must protect the environment variables from being substituted by their shell.

```
[user] $  export VMCATCHER_CACHE_EVENT="./myEventProcessor \$VMCATCHER_EVENT_TYPE"
```

An example of how to execute a command with an action command line.

VMCATCHER_LOG_CONF

Sets the path to the logging configuration file.

VMCATCHER_DIR_CERT

Sets the Path to the certificate authorities public keys, certificate revocation lists and certificate name spaces.

VMCATCHER_CACHE_DIR_CACHE

Path used by ’vmcatcher_endorser’ to store verified VM images.

VMCATCHER_CACHE_DIR_DOWNLOAD

Path used by ’vmcatcher_endorser’ to download VM images before VM image integrity is checked.

VMCATCHER_CACHE_DIR_EXPIRE

Path used by ’vmcatcher_endorser’ to store VM images when they are no longer endorsed.

VMCATCHER_CACHE_ACTION_DOWNLOAD

Instructs ’vmcatcher_endorser’ to download the latest VM images and check integrity.

VMCATCHER_CACHE_ACTION_CHECK

Instructs ’vmcatcher_endorser’ check integrity for all currently stored VM images.

VMCATCHER_CACHE_ACTION_EXPIRE

Instructs ’vmcatcher_endorser’ to expire stored VM images that are no longer endorsed.

vmcatcher_endorser

This application is for managing who the subscriber trusts to update image lists. Since individuals are identified with x.509 certificates, Each certificate has an issuing certificate and a unique string called a ’subject’ to identify the certificate. The ’subject’ of a certificate and the ’subject’ of the issuing certificate combined are called ’credentials’, and will be globally unique.

Individuals on rare occasions will need more than one certificate, for this reason they are given a unique identifier under this system and allowed to have more than one set of credentials.

Adding a individual to the vmcatcher database.

```
[user] $  vmcatcher_endorser --create \
       --endorser_uuid=e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
       --subject='/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=buncic/CN=379010/CN=Predrag Buncic'
 \
       --issuer='/DC=ch/DC=cern/CN=CERN Trusted Certification Authority'
```

Deleting and individual from a vmcatcher database.

```
[user] $  vmcatcher_endorser --delete \
       --endorser_uuid=e55c1afe-0a62-4d31-a8d7-fb8c825f92a2
```

Allowing an individual to update a subscription.

```
[user] $  vmcatcher_endorser --link \
       --endorser_uuid=e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
       --subscription_uuid=63175437-7d59-4851-b333-c96cb6545a86
```

Removing an individuals right to update a subscription.

```
[user] $  vmcatcher_endorser --unlink \
       --endorser_uuid=e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
       --subscription_uuid=63175437-7d59-4851-b333-c96cb6545a86
```

Each endorser_uuid must be unique or they will be assumed to be the same item. The endorser_uuid could be a more human name:

```
[user] $  vmcatcher_endorser --create \
       --endorser_uuid='Predrag Buncic' \
       --subject='/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=buncic/CN=379010/CN=Predrag Buncic'
 \
       --issuer='/DC=ch/DC=cern/CN=CERN Trusted Certification Authority'
```

vmcatcher_subscribe

This application manages your subscriptions and their update:

To add a subscription

```
[user] $  vmcatcher_subscribe  -s https://cernvm.cern.ch/releases/image.list
```

Or alternatively you can download a file visually inspect it and subscribe to the local file.

```
[user] $  vmcatcher_subscribe  -s file:////`pwd`/hepix_signed_image_list
```

To update your subscriptions

```
[user] $  vmcatcher_subscribe  -U
```

To list subscriptions

```
[user] $  vmcatcher_subscribe  -l
63175437-7d59-4851-b333-c96cb6545a86    True    https://cernvm.cern.ch/releases/image.list
```

Getting Information on a subscription:

```
[user] $  vmcatcher_subscribe  -i --uuid=63175437-7d59-4851-b333-c96cb6545a86
dc:identifier=63175437-7d59-4851-b333-c96cb6545a86
subscription.dc:description=CERN Virtual Machine
subscription.sl:authorised=True
subscription.hv:uri=https://cernvm.cern.ch/releases/image.list
subscription.dc:date:updated=2011-04-16T19:23:19Z
imagelist.dc:date:imported=2011-04-16T19:23:18Z
imagelist.dc:date:created=2011-03-16T00:15:07Z
imagelist.dc:date:expires=2011-04-13T00:15:07Z
```

you can also select on the basis of url:

```
[user] $  vmcatcher_subscribe  -i -r https://cernvm.cern.ch/releases/image.list
dc:identifier=63175437-7d59-4851-b333-c96cb6545a86
subscription.dc:description=CERN Virtual Machine
subscription.sl:authorised=True
subscription.hv:uri=https://cernvm.cern.ch/releases/image.list
subscription.dc:date:updated=2011-04-17T19:04:35Z
imagelist.dc:date:imported=2011-04-17T19:04:34Z
imagelist.dc:date:created=2011-03-16T00:15:07Z
imagelist.dc:date:expires=2011-04-13T00:15:07Z
```

Change the output format to get the original message without the security wrapper, or in original form:

```
[user] $  vmcatcher_subscribe  -i --uuid=63175437-7d59-4851-b333-c96cb6545a86 -f message
```

Three formats exist SMIME, message, lines.

- SMIME for applications that wish to process the signature as if from the endorser directly.
- message for applications that have no interest in processing the SMIME signature.
- lines for human users of this application.

To delete a subscription

```
[user] $  vmcatcher_subscribe  -D  --uuid=63175437-7d59-4851-b333-c96cb6545a86
```

vmcatcher_image

This application manages images within your subscription.

List the available images

```
[user] $  vmcatcher_image -l
327016b0-6508-41d2-bce0-c1724cb3d3e2    2       63175437-7d59-4851-b333-c96cb6545a86
858a817e-0ca2-473f-89d3-d5bdfc51968e    3       63175437-7d59-4851-b333-c96cb6545a86
da42ca85-179b-4873-b12e-32d549bf02b6    2       63175437-7d59-4851-b333-c96cb6545a86
```

The results show the `RFC 4122 UUID` of the image, the availability state and the subscription `RFC 4122 UUID`. The state value is a bitmap, 1 is subscribed, 2 means its available in the current `Virtual Machine Image List`s. Now we will select an image for local caching.

Selecting Images

Images can be selected by either `RFC 4122 UUID` or Sha512 hash. This allows explicit selection of images or by the sha512 from an old image.

Delete the subscription by image.

```
[user] $  vmcatcher_image -D -u 327016b0-6508-41d2-bce0-c1724cb3d3e2
```

Subscribe to an image.

```
[user] $  vmcatcher_image -a -u 327016b0-6508-41d2-bce0-c1724cb3d3e2
```

Unsubscribe an image

```
[user] $  vmcatcher_image -r -u 327016b0-6508-41d2-bce0-c1724cb3d3e2
```

vmcatcher_cache

This application downloads images. By default it will download images, check the sha512 hash of cached images and expire images from old `Virtual Machine Image Lists`.

```
[user] $  vmcatcher_cache
INFO:vmcatcher_subscribe.main:Defaulting DB connection to 'sqlite:///vmcatcher.db'
INFO:vmcatcher_subscribe.main:Defaulting actions as 'expire', 'sha512' and 'download'.
INFO:vmcatcher_subscribe.main:Defaulting cache-dir to 'cache'.
INFO:vmcatcher_subscribe.main:Defaulting partial-dir to 'cache/partial'.
INFO:vmcatcher_subscribe.main:Defaulting expired-dir to 'cache/expired'.
INFO:DownloadDir:Downloading '858a817e-0ca2-473f-89d3-d5bdfc51968e'.
INFO:CacheMan:moved file 858a817e-0ca2-473f-89d3-d5bdfc51968e
```

vmcatcher_cache Event interface

Since this application suite is intended to be embedded in a larger application and concerned with downloading and managing updates of VM images into a cloud infrastructure, it is some times beneficial to have an event interface so that applications may embed these applications in larger systems.

```
[user] $  vmcatcher_cache -x "/usr/bin/VmImageUpdateProcessor \$VMCATCHER_EVENT_TYPE"
```

The events interface launches a shell with a series of environment variables. The event must process its command within 10 seconds or else it will be sent a termination signal. See the following example:

```
[user] $  vmcatcher_cache -x 'env  ; exit 1'
```

All Events have a type. This is given to the event handler by setting the variable, `VMCATCHER_EVENT_TYPE` with the following values "AvailablePrefix", "AvailablePostfix", "ExpirePrefix" and "ExpirePosfix".

"Available" events happen when a new image is validated, while "Expire" events occur when an image i no longer the validated image. The "Prefix" events occur before the file changes state, and the "Posfix" events occur after the state change.

The following environment variables may be set by events:

- VMCATCHER_EVENT_TYPE
- VMCATCHER_EVENT_DC_DESCRIPTION
- VMCATCHER_EVENT_DC_IDENTIFIER
- VMCATCHER_EVENT_DC_TITLE
- VMCATCHER_EVENT_HV_HYPERVISOR
- VMCATCHER_EVENT_HV_SIZE
- VMCATCHER_EVENT_HV_URI
- VMCATCHER_EVENT_SL_ARCH
- VMCATCHER_EVENT_SL_CHECKSUM_SHA512
- VMCATCHER_EVENT_SL_COMMENTS

- VMCATCHER_EVENT_SL_OS
- VMCATCHER_EVENT_SL_OSVERSION
- VMCATCHER_EVENT_TYPE
- VMCATCHER_EVENT_FILENAME
- VMCATCHER_EVENT_IL_DC_IDENTIFIER

These correspond to the variables within the `Virtual Machine Image List`.

`vmcatcher_cache` Event Environment variables

VMCATCHER_EVENT_TYPE

- AvailablePrefix
  An image will be available soon as it is being attempted to be retrieved.
- AvailablePostfix
  An image was successfully validated as being available and placed in the cache directory.
- ExpirePrefix
  This image is will no longer be available in the cache directory.
- ExpirePosfix
  This image is no longer in the cache directory.

VMCATCHER_EVENT_DC_DESCRIPTION

The description text in the image.

VMCATCHER_EVENT_DC_IDENTIFIER

Unique identifier of the image. Its suggested that image producers use RFC 4122 `RFC 4122 UUID` for `Virtual Machine Image List` this allows updating the list, and uniqueness.

VMCATCHER_EVENT_DC_TITLE

Image Title.

VMCATCHER_EVENT_HV_HYPERVISOR

Typically set to reflect the Virtualization technology values such as "xen", "kvm".

VMCATCHER_EVENT_HV_SIZE

The Image Size

VMCATCHER_EVENT_HV_URI

The Original URI for the image

VMCATCHER_EVENT_SL_ARCH

The images architecture.

VMCATCHER_EVENT_SL_CHECKSUM_SHA512

The Images sha512 checksum.

VMCATCHER_EVENT_SL_COMMENTS

Comments added by the image author

VMCATCHER_EVENT_SL_OS

The Operating System the VM image contains

VMCATCHER_EVENT_SL_OSVERSION

The Operating System version

VMCATCHER_EVENT_FILENAME

The Image file name.

VMCATCHER_EVENT_IL_DC_IDENTIFIER

The image list the image comes from.

VMCATCHER_EVENT_HV_FORMAT

The format of the image. This is only available if the image list contains the format metadata.

Set up for Production using Cron

Then the by hand configuration for your master DB

```
[root] #  useradd vmcatcher
```

```
[root] #  mkdir -p /var/lib/vmcatcher /var/cache/vmimages/endorsed \
      /var/cache/vmimages/partial /var/cache/vmimages/expired
```

```
[root] #  touch /var/log/vmcatcher.log
```

```
[root] #  chown vmcatcher:vmcatcher /var/lib/vmcatcher  /var/cache/vmimages/endorsed \
      /var/cache/vmimages/partial /var/cache/vmimages/expired \
      /var/log/vmcatcher.log
```

```
[root] #  sudo -u vmcatcher /usr/bin/vmcatcher_subscribe \
      -s https://cernvm.cern.ch/releases/image.list \
      -d sqlite:////var/lib/vmcatcher/vmcatcher.db
```

make a cron job

```
[root] #  cat   /etc/cron.d/vmcatcher
export VMCATCHER_RDBMS="sqlite:////var/lib/vmcatcher/vmcatcher.db"
export VMCATCHER_CACHE_DIR_CACHE="/var/cache/vmimages/endorsed/"
export VMCATCHER_CACHE_DIR_DOWNLOAD="/var/cache/vmimages/partial/"
export VMCATCHER_CACHE_DIR_EXPIRE="/var/cache/vmimages/expired/"
export VMCATCHER_CACHE_EVENT="python /usr/share/doc/vmcatcher-0.1.29/vmcatcher_eventHndlExpl  --
output_file=/tmp/foo --datetime"
```

```
50 */6 * * * vmcatcher (/usr/bin/vmcatcher_subscribe -U; /usr/bin/vmcatcher_cache  ) >> /var/log/
vmcatcher.log 2>&1
```

So the script is executed every 6 hours shortly after fetch CRL.

If a new `Virtual Machine Image` is downloaded, or an old `Virtual Machine Image` is expired the event will trigger `VMCATCHER_CACHE_EVENT` and the application `vmcatcher_eventHndlExplscript` will append the data to /tmp/foo

Now at any time users with file permissions can get a list of valid images.

```
[user] $ VMCATCHER_RDBMS="sqlite:////var/lib/vmcatcher/vmcatcher.db" vmcatcher_image -l
```

Replacing the event handler

`vmcatcher_cache` produces "events" in the form of launching an application when a new image is downloaded or expired, this application is then launched with environment variables that include the image ID,date, etc etc. Since all the information about current images is available from other vmcatcher commands you don’t need to handle events but it does make it simpler for some setups.

So with a cron job like:

```
[root] #  cat   /etc/cron.d/vmcatcher
export VMCATCHER_RDBMS="sqlite:////var/lib/vmcatcher/vmcatcher.db"
export VMCATCHER_CACHE_DIR_CACHE="/var/cache/vmimages/endorsed/"
export VMCATCHER_CACHE_DIR_DOWNLOAD="/var/cache/vmimages/partial/"
export VMCATCHER_CACHE_DIR_EXPIRE="/var/cache/vmimages/expired/"
export VMCATCHER_CACHE_EVENT="python /usr/share/doc/vmcatcher-0.1.29/vmcatcher_eventHndlExpl  --
output_file=/tmp/foo --datetime"

50 */6 * * * vmcatcher (/usr/bin/vmcatcher_subscribe -U; /usr/bin/vmcatcher_cache  ) >> /var/log/
vmcatcher.log 2>&1
```

The `VMCATCHER_CACHE_EVENT` environment variable specifies the even handler.

It is expected that `vmcatcher_eventHndlExplscript` will be replaced by sites wanting to load images into image catalogues of popular clouds. "vmcatcher_eventHndlExplscript" is an example "event" handler, which takes a path parameter. It reads the environment variables generates a simple JSON output line, and appends it to the file described in the path parameter.

It is recommended the replacement `vmcatcher_eventHndlExplscript` copies the images from the `VMCATCHER_CACHE_DIR_CACHE` directory to the cloud, as `vmcatcher_cache` assumes it can delete and update its local cache. A second recommendation for site replacement is that `vmcatcher_eventHndlExplscript` should do very little and end quickly as then `vmcatcher_cache` can process the next download without blocking. Since events are not resent so error handling is more complex, it may be wise to use a message queue, or storing the event and processing after, rather than just using a simple fork.

Logging configuration

All scripts have a logging option. This is used to configure pythons logging library. An example is shown below.

```
[user] $  vmcatcher_image -L /usr/share/doc/vmcatcher/logger.conf -l
```

Logging can be independently set up for each object to multiple locations, and with different log levels.

To Do (16-05-2012)

- Only message authenticity is checked, does not yet check authenticity of transport.
- PGP signatures.
- Support encrypted messages.

While it does check the authenticity of the message using `x.509`, at the moment the authenticity of the host is unchecked. For the ease of programing it would be far simpler to use `x.509` certificates to check the host server. In terms of deployment it would be far easier just to check any host key mechanism, as this is sufficient.