Amazon EC2

Developer Guide

Amazon EC2: Developer Guide

Table of Contents

Introduction

Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Developer Guide.

This guide picks up where the Getting Started Guide ends and will provide you with the information necessary for creating more sophisticated AMIs, using advanced service features, and writing applications using Amazon EC2. This guide assumes you have worked through the Getting Started Guide, installed the command line and API tools as described, and have a general understanding of the service.

The chapters presented in the guide are:

- *Working with AMIs* walks you through the steps required to create the customized package of software that will execute on your host - essentially packaging your desired Operating System configuration.
- *Launching and Using Instances* provides an overview of the Amazon EC2 instances and some tips for using them effectively.
- *Securing the Network* provides an overview of the distributed firewall and usage examples.
- *Using the APIs* explains the basics of using the SOAP and Query APIs, including signing requests.
- *API Reference* provides a comprehensive reference to the SOAP and Query APIs.
- *Command Line Tools Reference* provides a comprehensive reference to the command line tools supplied by Amazon EC2.
- *Technical FAQ* is a collection of interesting and commonly asked questions.
- *Glossary* is a simple glossary of Amazon EC2 terminology.

Working with AMIs

This section details how to build, store and share AMIs.

Creating an AMI

There are several techniques for creating an AMI offering a mix of ease of use and detailed customization levels. The easiest method involves starting from an existing public AMI and modifying it according to your requirements, as described in the section called "Starting with an Existing AMI".

Another approach is to build a fresh installation either on a stand-alone machine or on an empty file system mounted by loopback. This essentially entails building an operating system installation from scratch and is described in the section called "Creating via a Loopback File".

Once the installation package has been built to your satisfaction it needs to be bundled and uploaded to Amazon S3 as described in the section called "Bundling an AMI".

Starting with an Existing AMI

This is the quickest and easiest of the methods to get a new working AMI. Start with an existing public AMI or one of your own. You can then modify that as you see fit and subsequently create a new AMI with the **ec2-bundle-vol** utility, as decribed later in the section called "Bundling an AMI".

Select an AMI

The first step is to locate an AMI that contains the packages and services that you require. This can be one of your own AMIs or one of the public AMIs provided by Amazon EC2. Use **ec2-describe-images** to get a list of available AMIs, as is shown below, then select one of the listed AMIs and note its AMI ID, e.g. ami-5bae4b32:

```
PROMPT>  ec2-describe-images
IMAGE ami-60a54009 ec2-public-images/base-fc4-apache.manifest.xml
475219833042 available public
IMAGE ami-61a54028 <your-s3-bucket>/image.manifest.xml 495219933132 available
private
IMAGE ami-5bae4b32 ec2-public-images/getting-started.manifest.xml
475219833042 available public
IMAGE ami-6ea54007 ec2-public-images/base-fc3-mysql.manifest.xml 475219833042
available public
```

Generate a Keypair

This step is only required if you've selected one of the public AMIs provided by Amazon EC2. A public/private keypair must be created to ensure that you, and only you, have access to the instances that you launch.

```
PROMPT>  ec2-create-keypair gsg-keypair
KEYPAIR gsg-keypair
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBuLFg5ujHrtm1jnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFxTHgElQiJLChp
HungXQ29VTc8rc1bW0lkdi23OH5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKCOVUR+jx71Sg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjGlUKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3LpGZyq/
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYGtGSBMpTRP5hnbzzuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAY1tsiUsIwDl5
91CXirkYGuVfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVD1pM
ZNUJs7rw9gZRTrf7LylaJ58kOcyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1kGQI7
3wcfgt5ecIu4TZf0OE9IHjn+2eRlsrjBdeORi7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhHOux5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLGCwU9h9HlO9mKAc2m8Cm1
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkpHi2EISh1uRkhxljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAy1/EDNTY5OSIJU+0KFmQbyhsbm
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfqGoQc7+sAc8vmKkesnF7CqCEKDyF/dhrxYdQKB
```

```
gC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeEO+uZ58/BGKOIGHByHBDiXtzMhdJr15HTYjxK7OgTZm
gK+8zp4L9IbvLGDMJO8vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+Zrvm1F0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0gij1houRe
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----
```

The resulting private key must be saved in a local file for later use. Create a file named `id_rsa-gsg-keypair` and paste into it all lines starting with the line "`-----BEGIN PRIVATE KEY-----`" and ending with the line "`-----END PRIVATE KEY-----`". Confirm that the file contents looks exactly as shown below.

```
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBuLFg5ujHrtm1jnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFxTHgElQiJLChp
HungXQ29VTc8rc1bW0lkdi23OH5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKCOVUR+jx71Sg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjGlUKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3LpGZyq/
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYGtGSBMpTRP5hnbzzuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAY1tsiUsIwDl5
91CXirkYGuVfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVD1pM
ZNUJs7rw9gZRTrf7LylaJ58kOcyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1kGQI7
3wcfgt5ecIu4TZf0OE9IHjn+2eRlsrjBdeORi7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhHOux5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLGCwU9h9HlO9mKAc2m8Cm1
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkpHi2EISh1uRkhxljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAy1/EDNTY5OSIJU+0KFmQbyhsbm
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfqGoQc7+sAc8vmKkesnF7CqCEKDyF/dhrxYdQKB
gC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeEO+uZ58/BGKOIGHByHBDiXtzMhdJr15HTYjxK7OgTZm
gK+8zp4L9IbvLGDMJO8vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+Zrvm1F0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0gij1houRe
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----
```

Launch an Instance

You are now ready to launch an instance of the AMI you selected above.

```
PROMPT>  ec2-run-instances ami-5bae4b32 -k gsg-keypair
INSTANCE        i-10a64379   ami-5bae4b32   EC2    pending   gsg-keypair  0
```

The instance ID in the second field of the output is a unique identifier for the instance and can be used subsequently to manipulate your instance, e.g. to terminate it.

> **Important**
> Once you launch an instance, you will be billed per hour for CPU time. Make sure you terminate any instances which you don't intend to leave running indefinitely.

It will take a few minutes for the instance to launch. You can follow its progress by running:

```
PROMPT>  ec2-describe-instances i-10a64379
RESERVATION     r-fea54097  495219933132   EC2
INSTANCE        i-10a64379  ami-5bae4b32   domU-
12-34-31-00-00-05.usma1.compute.amazonaws.com   EC2    running   gsg-keypair
0
```

When the status field reads "running", the instance has been created and has started booting. There may still be a short time before it is accessible over the network, however. The DNS name displayed in the sample output above will be different from that assigned to your instance. Make sure you use the

appropriate one.

Authorize Network Access

In order to be able to reach the running instance from the Internet, you need to enable access for the ssh service which runs on port 22:

```
PROMPT>  ec2-authorize default -p 22
PERMISSION     default  ALLOWS  tcp     22      22      FROM    CIDR
0.0.0.0/0
```

Connect to the Instance

Now that you have a running instance, you can log in and modify it according to your requirements. If you launched a public Amazon EC2 AMI, you can use the following command to log in with your own private key:

```
PROMPT>  ssh -i id_rsa-gsg-keypair root@domU-
12-34-31-00-00-05.usma1.compute.amazonaws.com
root@my-instance #
```

Otherwise, use the plain ssh command and supply the appropriate password when prompted.

```
PROMPT>  ssh root@domU-12-34-31-00-00-05.usma1.compute.amazonaws.com
root@my-instance #
```

You now have complete control over the instance and may add, remove, modify or upgrade packages and files to suit your needs. Some of the basic configuration settings related to the Amazon EC2 enviroment, such as the network interface configuration and `/etc/fstab` contents, should only be changed with extreme care, to avoid making the AMI unbootable or inaccessible from the network once running.

Upload the Key and Certificate

The new AMI will be encrypted and signed to ensure that it can only be accessed by you and Amazon EC2. You therefore need to upload your Amazon EC2 private key and X.509 certificate to the running instance, for use in the AMI bundling process.

Assuming the private key and X.509 certificate are contained in files `pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem` and `cert-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem`, copy both of these files to your instance:

```
PROMPT> scp pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem cert-HKZYK-
TAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem root@domU-
12-34-31-00-00-05.usma1.compute.amazonaws.com:/tmp
pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem                          100%  717
0.7KB/s   00:00 cert-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem
100%  685     0.7KB/s   00:00
```

> **Note**
> It is important that the key and cert files are uploaded into /tmp to prevent them being bundled with the new AMI.

You are now ready to proceed to the next step which involves bundling the volume and uploading the resulting AMI to Amazon S3. This is described in the section called “Bundling an AMI”.

Creating via a Loopback File

This method entails doing a full operating system installation on a clean root file system, but avoids having to create a new root disk partition and file system on a physical disk. Once you have installed your operating system, the resulting image can be bundled as an AMI with the **ec2-bundle-image** utility.

Create a File to Host the AMI

The **dd** utility can be used to create files of arbitrary sizes. In this case, make sure to create a file large enough to host the operating system, tools and applications that you will install. For example, a baseline Linux installation requires about 700MB, so your file should be at least 1GB. The command below creates a file of 1024*1MB=1GB.

```
# dd if=/dev/zero of=my-image.fs bs=1M count=1024
1024+0 records in
1024+0 records out
```

Create a Root File System Inside the File

There are several variations on the generic **mkfs** utility that can be used to create a file system inside `my-image.fs`. Typical Linux installations default to `ext2` or `ext3` file systems. Create an `ext3` file system by issuing the following command:

```
# mke2fs -F -j my-image.fs
mke2fs 1.38 (30-Jun-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
131072 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
16384 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 24 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Mount the File via Loopback

The loopback module allows you to use a normal file as if it were a raw device. In this manner you get a file-system in a file. Mounting a file system image file via loopback presents it as part of the normal file system. You can then modify it using your favourite file management tools and utilities. Create a mount point in the file system where the image will be attached and then mount the file system image, as follows:

```
# mkdir /mnt/ec2-fs
# mount -o loop my-image.fs /mnt/ec2-fs
```

Prepare for the Installation

Before the operating system installation can proceed, some basic files have to be created and prepared on the newly created root file system.

Create `/dev`

Create a `/dev` directory and populate it with a minimal set of devices (you can ignore the errors in the output):

```
# mkdir /mnt/ec2-fs/dev
# /sbin/MAKEDEV -d /mnt/ec2-fs/dev -x console
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
# /sbin/MAKEDEV -d /mnt/ec2-fs/dev -x null
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
# /sbin/MAKEDEV -d /mnt/ec2-fs/dev -x zero
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
MAKEDEV: mkdir: File exists
```

Create `/etc`

Create an `/etc` directory:

```
# mkdir /mnt/ec2-fs/etc
```

Create `/mnt/ec2-fs/etc/fstab` and add the following entries to it:

```
/dev/sda1  /         ext3    defaults        1 1
none       /dev/pts  devpts  gid=5,mode=620  0 0
none       /dev/shm  tmpfs   defaults        0 0
none       /proc     proc    defaults        0 0
none       /sys      sysfs   defaults        0 0
```

Create `yum-xen.conf`

Create a temporary **yum** configuration file that will ensure all the required basic packages and utilities are installed. This configuration file can be created anywhere on your main file system, but for now we'll assume that you create it in your working directory. Just to clarify, it does not need to be created in the loopback file system. It is used only during installation of the loopback file system. Create `yum-xen.conf` with the following content:

```
[main]
cachedir=/var/cache/yum
debuglevel=2
logfile=/var/log/yum.log
exclude=*-debuginfo
gpgcheck=0
obsoletes=1
reposdir=/dev/null

[base]
name=Fedora Core 4 - $basearch - Base
mirrorlist=http://fedora.redhat.com/download/mirrors/fedora-core-$releasever
enabled=1

[updates-released]
name=Fedora Core 4 - $basearch - Released Updates
mirrorl-
ist=http://fedora.redhat.com/download/mirrors/updates-released-fc$releasever
```

```
enabled=1
```

Mount `proc`

Due to a bug in the **groupadd** utility from the `shadow-utils` package (versions prior to 4.0.7-7), the new `proc` file system needs to be mounted by hand at this point.

```
# mkdir /mnt/ec2-fs/proc
# mount -t proc none /mnt/ec2-fs/proc
```

Install the Operating System

At this stage all the basic directories and files have been created and you are ready to do the operating system installation. This process might take a while depending on the speed of the host and the network link to the repository.

```
# yum -c yum-xen.conf --installroot=/mnt/ec2-fs -y groupinstall Base
Setting up Group Process
Setting up repositories
base                      100% |=========================| 1.1 kB    00:00
updates-released          100% |=========================| 1.1 kB    00:00
comps.xml                 100% |=========================| 693 kB    00:00
comps.xml                 100% |=========================| 693 kB    00:00
Setting up repositories
Reading repository metadata in from local files
primary.xml.gz            100% |=========================| 824 kB    00:00
base      : ################################################## 2772/2772
Added 2772 new packages, deleted 0 old in 15.32 seconds
primary.xml.gz            100% |=========================| 824 kB    00:00
updates-re: ################################################## 2772/2772
Added 2772 new packages, deleted 0 old in 10.74 seconds
...
Complete!
```

Congratulations!

You now have a base installation in the image file you've created. The next steps are to configure the installation to operate inside Amazon EC2, and to customize the installation for your use.

Configure the Installed Operating System

The base operating system has now successfully been installed. You must now configure the networking and hard drives to work in the Amazon EC2 environment.

Configure the Network Interface

The Amazon EC2 environment provides a networking interface card that needs to be configured to provide external network access for the running instance. Edit (or create) the following file `/mnt/ec2-fs/etc/sysconfig/network-scripts/ifcfg-eth0`, making sure it contains at least the following information.

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
TYPE=Ethernet
USERCTL=yes
PEERDNS=yes
IPV6INIT=no
```

Note
The Amazon EC2 DHCP server ignores hostname requests. If you set `DHCP_HOSTNAME` the local hostname will be set on the instance but not externally. In addition, this local hostname will be the same for all instances of the AMI, which may prove confusing.

Enable Networking

After configuring the network interface, you need to ensure that networking will come up when the system is started. To do this, ensure that (at least) the following appears in `/mnt/ec2-fs/etc/sysconfig/network`.

```
NETWORKING=yes
```

Set up Hard Drives in `/etc/fstab`

Amazon EC2 provides the instance with additional local storage by way of a disk drive on `/dev/sda2`. In addition, swap space is provided on `/dev/sda3`. To ensure both these are mounted at system start up time, add the following lines to `/mnt/ec2-fs/etc/fstab`:

```
/dev/sda2  /mnt      ext3    defaults        1 2
/dev/sda3  swap      swap    defaults        0 0
```

Configure Additional Services

Finally, make sure that all of your required services will be started at system start up time by allocating them to the appropriate system run levels. To enable the service `my-service` on multi-user and networked run levels, for example, execute:

```
# chroot /mnt/ec2-fs /bin/sh
# chkconfig --level 345 my-service on
# exit
```

Unmount the Loopback File

Your new installation has now been successfully installed and configured to operate in the Amazon EC2 environment. You may now unmount the image:

```
# umount /mnt/ec2-fs/proc
# umount -d /mnt/ec2-fs
```

Bundling an AMI

A root file system image needs to be bundled as an AMI in order to be used with the Amazon EC2 service. The bundling process first compresses the image to minimize bandwidth usage and storage requirements. The compressed image is then encrypted and signed to ensure confidentiality of the data, and authentication against the creator. The encrypted image is finally split into manageable parts for upload. A manifest file is created containing a list of the image parts with their checksums. This chapter provides an overview of the AMI tools that automate this process and some examples of their use.

The AMI tools are three command-line utilities:

1. **ec2-bundle-image** bundles an existing AMI
2. **ec2-bundle-vol** creates an AMI from an existing machine or installed volume
3. **ec2-upload-bundle** uploads a bundled AMI to S3 storage

Installing the AMI Tools

The AMI tools are packaged as an RPM suitable for running on Fedora Core 3/4 with Ruby 1.8.2 (or greater) installed. On Fedora Core 4 Ruby can be installed by following the steps below. You will need root privileges to install the software. You can find the AMI tools RPM from our public S3 downloads bucket.

First install Ruby using the yum package manager.

```
# yum install ruby
```

Install the AMI tools RPM.

```
# rpm -i ec2-ami-tools-x.x-xxxx.i386.rpm
```

Installation Issues

The AMI tools libraries install under `/usr/lib/site_ruby`. Ruby should pick up this path automatically, but if you see a load error when running one of the AMI utilities, it may be because Ruby isn't looking there. To fix this, add `/usr/lib/site_ruby` to Ruby's library path, which is set in the `RUBYLIB` environment variable.

Documentation

The manual describing the operation of each utility can be displayed by invoking it with the `--manual` parameter. For example:

```
# ec2-bundle-image --manual
```

Invoking a utility with the `--help` parameter displays a summary and list of command line parameters. For example:

```
# ec2-bundle-image --help
```

Using the AMI Tools

Once a machine image has been created it must be bundled as an AMI for use with Amazon EC2, as follows. Use **ec2-bundle-image** to bundle an image that you have prepared in a loopback file, as described in the previous section.

```
# ec2-bundle-image -i my-image.img -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem
-c cert-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem -u 12345678
```

This will create the bundle files:

```
image.part.00
image.part.01
...
image.part.NN
image.manifest.xml
```

Alternatively an AMI could be created by snapshotting the local machine root file system and bundling it all at once by using **ec2-bundle-vol**. (note: you will need to have root privileges to do this and SELinux must be disabled). Use **ec2-bundle-vol** to re-bundle a (modified) running instance of an existing AMI, as described in the previous section.

```
# ec2-bundle-vol -k pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem -c cert-
HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem -s 1000 -u 495219933132
```

As with **ec2-bundle-image**, **ec2-bundle-vol** will create image parts files and a manifest file.

Note

If selinux is enabled when **ec2-bundle-vol** is run, the filesystem creation step may fail. Selinux should be disabled while this is done.

Uploading a Bundled AMI

The bundled AMI needs to be uploaded for storage in Amazon S3 before it can be accessed by Amazon EC2. Use **ec2-upload-bundle** to upload the bundled AMI that you created as described above. S3 stores data objects in buckets, which are similar in concept to directories. Buckets must have globally unique names. The **ec2-upload-bundle** utility will upload the bundled AMI to a specified bucket. If the specified bucket does not exist it will be created. However, if the specified bucket already exists, and belongs to another user, then **ec2-upload-bundle** will fail.

```
# ec2-upload-bundle -b my-bucket -m image.manifest.xml -a my-
aws-access-key-id -s my-secret-key-id
```

The AMI manifest file and all image parts are uploaded to S3. The manifest file is encrypted with the Amazon EC2 public key before being uploaded.

Building Shared AMIs

This section describes best practices for building shared AMIs. Building safe, secure, useable AMIs for public consumption is a fairly straightforward process, if you stick to a few simple guidelines.

You're welcome to choose to ignore any, or all, of these guidelines. They're not requirements for publishing an AMI. However, we believe that following these guidelines will make for a far smoother user experience and help ensure your users' instances are secure.

Platform Notes

These guidelines are generally written with Fedora distros in mind, but the principles hold for any AMI. You may need to tweak the examples we've provided to get them to work on other distributions.

Many of the steps below involve automating something during the boot sequence. We've made a few notes for some of the more common distros below. For other distros check your local documentation or search the AWS forums in case someone else has done it already.

- On Red Hat and Fedora systems you can add these steps to your `/etc/rc.d/rc.local` script.
- On Gentoo systems you can add them to `/etc/conf.d/local.local`.
- On Ubuntu systems you can add them to `/etc/rc.local`.
- On Debian, you may need to create a start up script in `/etc/init.d` and use `update-rc.d <scriptname> defaults 99` (where `<scriptname>` is the name of the script you created) and add the steps to this script.

Update the AMI Tools at Boot Time.

We recommend that during the boot process your AMIs should fetch and upgrade the EC2 AMI creation tools. This ensures that new AMIs based on your shared AMIs contain the latest AMI creation tools.

On Fedora, adding the following to `rclocal` will update the AMI tools at boot.

```
# Update the EC2 AMI creation tools
echo " + Updating ec2-ami-tools"
wget http://s3.amazonaws.com/ec2-downloads/ec2-ami-tools.noarch.rpm && \
rpm -Uvh ec2-ami-tools.noarch.rpm && \
echo " + Updated ec2-ami-tools"
```

You may wish to use this pattern to auto update other software on your image. It's up to you to decide which, if any, of the software components installed on your AMI should be updated at boot time. Two things to consider when making this decision are how much WAN traffic will the update generate (bearing in mind your users will be charged for it) and how much risk is there that the update will break other software on the AMI.

Disable Password Based Logins for Root

A fixed root password for a public AMI is a security risk. It won't be long before it becomes well known. It's not sufficient to rely on users changing the password after logging in for the first time, since this leaves a small window of "opportunity" for someone looking for a chance to do something bad (or cheap thrills).

The solution is to disable password based logins for the root user. In fact, we recommend you go one step further and randomize the root password at boot, just in case. Defense-in-depth is always a good strategy.

To disable password based logins for root, edit the `/etc/ssh/sshd_config` file and find and change the following line

```
#PermitRootLogin yes
```

to

```
PermitRootLogin without-password
```

The location of this configuration file may differ for your distribution, or if you're not running OpenSSH. Consult the relevant documentation if this is the case.

Randomizing the root password is also pretty simple. Add the following to your boot process.

```
if [ -f "/root/firstrun" ] ; then
  dd if=/dev/urandom count=50|md5sum|passwd --stdin root
  rm -f /root/firstrun
else
  echo "* Firstrun *" && touch /root/firstrun
fi
```

Once again, you may need to consult the relevant documentation if you're using a distro other than Fedora.

Install Public Key Credentials.

Now that we've done a pretty thorough job of ensuring that no one can log into instances of our AMI using a password, we need to make sure they can login using some other mechanism.

EC2 allows users to specify a public-private keypair name when launching an instance. When a valid keypair name is provided to the RunInstances API call (or via the command line API tools) the following happens behind the scenes:

The public key (the only portion of the keypair EC2 retains on the server after a call to CreateKeyPair) is made available to the instance via two methods

1. an HTTP query
2. a file on the instance's ephemeral store (`/dev/sda2`). This file is named `openssh_id.pub` and its format is compatible with the OpenSSH `authorized_keys` file.

Note

The HTTP request is the *preferred method* of retrieving the public key. The second method is deprecated and will be phased out in future versions of the service.

This means at boot, all your AMI need do is retrieve the key value and append it to `/root/.ssh/authorized_keys` (or the equivalent for any other user account on the AMI) and users will be able to launch instances of your AMI with a keypair and log in without requiring a root password.

```
if [ ! -d /root/.ssh ] ; then
        mkdir -p /root/.ssh
        chmod 700 /root/.ssh
fi
# Fetch public key using HTTP
curl http://169.254.169.254/1.0//meta-data/public-keys/0/openssh-key > /
tmp/my-key
```

```
if [ $? -eq 0 ] ; then
        cat /tmp/my-key >> /root/.ssh/authorized_keys
        chmod 600 /root/.ssh/authorized_keys
        rm /tmp/my-key
fi
# or fetch public key using the file in the ephemeral store:
if [ -e /mnt/openssh_id.pub ] ; then
        cat /mnt/openssh_id.pub >> /root/.ssh/authorized_keys
        chmod 600 /root/.ssh/authorized_keys
fi
```

This can be applied to any user account. There is no reason to restrict it to root.

Note
There's an implication of this step that you should be aware of: rebundling an instance based on this image will include the key it was launched with in the new image, unless you explicitly clear out (or delete) the `authorized_keys` file. You can also exclude this file from rebundling.

Disable sshd DNS Checks

This is an optional step. It slightly weakens your sshd security (although not significantly), but ensures that should DNS resolution fail, ssh logins will still work. If you leave this setting at its default, DNS resolution failures will prevent logins altogether.

To disable password based logins for root, edit the `/etc/ssh/sshd_config` file and find and change the following line

```
#UseDNS yes
```

to this

```
UseDNS no
```

The location of this configuration file may differ for your distribution, or if you're not running OpenSSH. Consult the relevant documentation if this is the case.

Identify Yourself

Currently there is no easy way of knowing who provides a shared AMI. All you are presented with is a numeric user id. We suggest that you post a description of your ami, and the ami id, in the Amazon EC2 developer forum. This will provide users interested in trying new shared AMIs with a central location to find information about those AMIs.

We are working on making it easier to share and find new AMIs.

Protect Yourself

We have looked at making shared AMIs safe, secure and useable for the users who launch them, but if you publish a shared AMI you should also take steps to protect yourself against the users of you AMI This section looks at steps you can take to do this.

We recommend against storing sensitive data or software on any AMI that you share. Users who launch a shared AMI potentially have access to rebundle it and register it as their own. Follow these guidelines to help you to avoid some easily overlooked security risks:

- Always delete the shell history before bundling. If you attempt more than one bundle upload in the same image the shell history will contain your secret access key.
- Bundling a running instance requires your private key and X509 certificate. Put these and other credentials in a location that will not be bundled (such as the ephemeral store).
- Exclude the ssh authorized keys when bundling the image. The Amazon public images store the public key an instance was launched with in that instance's ssh authorized keys file.

It is not possible for this list to be exhaustive. Build your shared AMIs carefully and consider where you might be exposing sensitive data.

Sharing AMIs

Introduction

Amazon EC2 makes it possible for users to share their AMIs with other users. This section describes how to do this using the Amazon EC2 command line tools.

Please be sure to read the section called "Building Shared AMIs" (which highlights the security considerations of sharing AMIs) before proceeding.

AMIs have a `launchPermission` property that controls which users, besides the owner, are allowed to launch instances of that AMI. By modifying an AMI's `launchPermission` property it is possible to allow all users to launch the AMI (make the AMI public) or to allow only a few specific users to launch the AMI (explicit launch permissions).

The `launchPermission` attribute is a list of users and launch groups. Launch permissions can be granted by adding items to the list and revoked by removing items from the list. Explicit launch permissions for users are granted or revoked by respectively adding or removing their AWS account ids. The only launch group currently supported is the `all` group, which gives launch permissions to all users and makes the AMI public. In the rest of this chapter we refer to launch groups simply as groups. These launch groups are not the same as security groups and the two should not be confused. An AMI may have both public and explicit launch permissions.

The owner of an AMI is not billed when their AMI is launched by another user. Only the user launching the AMI is billed.

Making an AMI Public

An AMI is made public by adding the `all` group to the AMI's `launchPermission` attribute. This can be done with the ec2-modify-image-attribute command.

```
PROMPT>  ec2-modify-image-attribute ami-5bae4b32 --launch-permission -a all
launchPermission        ami-5bae4b32    ADD     group   all
```

To check the launch permissions on an AMI use the ec2-describe-image-attribute command. In this example the shortened form of **--launch-permission**, **-l**, is used.

```
PROMPT> ec2-describe-image-attribute ami-5bae4b32 -l
launchPermission        ami-5bae4b32    group   all
```

An AMI is be made private again by removing the `all` group from its launch permissions. This will not affect any explicit launch permissions the AMI may have or any running instances of the AMI.

```
PROMPT> ec2-modify-image-attribute ami-5bae4b32 -l -r all
launchPermission        ami-5bae4b32    REMOVE  group   all
```

Sharing an AMI with Specific Users

It is possible to share an AMI with specific users without making the AMI public. This is done by adding explicit launch permissions. To do this you need the user's AWS account id.

```
PROMPT> ec2-modify-image-attribute ami-5bae4b32 -l -a 495219933132
launchPermission        ami-5bae4b32    ADD     userId  495219933132
```

Explicit launch permissions are removed in the same way as public launch permissions.

```
PROMPT> ec2-modify-image-attribute ami-5bae4b32 -l -r 495219933132
launchPermission        ami-5bae4b32    REMOVE  userId  495219933132
```

Another way to remove launch permissions is to use the ec2-reset-image-attribute command. This will remove any launch permissions that have been added to an AMI, public and explicit. Owners always have launch permissions for their AMIs and will not lose those permissions by using `ec2-reset-image-attribute`.

```
PROMPT> ec2-reset-image-attribute ami-5bae4b32 -l
launchPermission        ami-5bae4b32    RESET
```

Publishing Shared AMIs

AMIs can be published by posting them in the Amazon Web Services Resource Center, Public AMIs Folder.

The following information must be included when publishing AMIs:

- AMI id
- AMI manifest

We recommend the following information should also be included when publishing AMIs:

- Publisher
- Publisher URL
- OS / Distribution
- Key Features
- Description
- Daemons / Services
- Release Notes

The following template can be cut and pasted into the document. You must be in HTML edit mode.

```
<strong>AMI ID: </strong>[ami-id]<br />
<strong>AMI Manifest: </strong>[bucket/image.manifest.xml]<br />
<h2>About this AMI</h2>
<ul>

    <li>Published by [Publisher] (<a
href="http://www.mysite.com">[http://www.mysite.com]</a>).<br />
            </li>
    <li>[Key Features] <br />
            </li>
    <li>[Description]</li>
    <li>This image contains the following daemons / services:
    <ul>

        <li>[Daemon 1]</li>
        <li>[Daemon 2]</li>
    </ul>
            </li>
</ul>
<h2><strong>What&#39;s New?</strong></h2>The following changes were made on
[Date].<br />
<ul>

    <li>[Release Notes 1]</li>
```

```
</ul>
<span style="font-size: x-small; font-family: courier
new,courier">     - [Note 1]</span><br />
<span style="font-size: x-small; font-family: courier
new,courier">     - [Note 2]</span><br />
<span style="font-size: x-small; font-family: courier
new,courier">     - [Note 3]</span><br />
<ul>
```

Launching and Using Instances

This section details how to launch instances and retrieve instance specific data from within the image. It also covers launching shared AMIs and security risks associated with running shared AMIs.

Using Instances

The instance is your basic computation building block. It is a medium-sized host that provides you with the same predictable performance you would expect from a physical host. You can run on as many or as few as you need at any given time. Each instance predictably provides the equivalent of a system with a 1.7Ghz x86 CPU, 1.75GB of RAM, 160GB of local disk, and 250Mb/s of network bandwidth.

Once launched, an instance looks very much like a traditional host. You have complete control of your instances. You have root access to each one, and you can interact with them as you would any machine.

Best Practices

Here are some suggestions for making the best use of the Amazon's EC2 instances.

- Do not rely on an instance's local storage for valuable, long-term data. Instances can fail, and when they fail, the data on the local disk is lost. You should use a replication strategy across multiple instances to keep your data safe or store your persistent data in Amazon S3.
- Define images based on the type of work your instances perform. For "internet applications" you may choose to define one image for database instances and one image for your webservers. Image creation and storage are cheap and easy operations. Individualize and customize as necessary. Keeping your images specialized will mean that the resulting AMIs can be smaller. Smaller AMIs will boot considerably faster.
- Monitor the health of your instances. Make your instances work for you by monitoring each other. You may choose to create an image which contains one of the various open-source monitoring tools such as Nagios or OpenNMS. Each worker instance, based on your other images, might then report its health to your monitoring instance.
- Keep your Amazon EC2 firewall permissions as restrictive as possible. Only open up permissions you need to open. Use separate groups to deal with instances that have different network ingress requirements. Consider using additional security measures inside your instance including your own firewall. If you need to login interactively (ssh), consider creating a bastion security group that allows external login, while the remainder of your instances are in a group that does not allow external login.

Using Instance Data

Introduction

Amazon EC2 instances may access instance-specific metadata as well as data supplied when launching the instances. This data can be used to build more generic AMIs (e.g. behavior could be modified by configuration files supplied at launch time).

Example Scenario

Perhaps you run web servers for various Mom-and-Pop stores. All the instances use the same AMI. At launch time you could specify which Amazon S3 bucket the AMI should retrieve its content from. This allows you to launch multiple Mom-and-Pop sites serving different content using the same AMI by doing the following:

- Create an Amazon S3 bucket
- Place your content in the Amazon S3 bucket
- Launch an instance of your web server AMI specifying the Amazon S3 bucket containing the web content

Categories of Available Data

The data available to instances is categorized into

metadata
This data is specific to an instance. Currently we provide:
- AMI id
- AMI manifest path
- AMI launch index
- Instance id
- Hostname
- Local IPv4 address
- Public keys (if supplied at instance launch time)
- Reservation id
- Security group names (if supplied at instance launch time)

user-supplied data
Any user-supplied data is treated as opaque data: what you give us is what you get back.

Note

- All instances launched together get the same user-supplied data. You may use the `AMI launch index` metadatum as an index into the data (example).
- User data is limited to 16K.
- The user data must be base64-encoded before being submitted to the API. The API command-line tools perform the base64-encoding for you. The data will be base64 decoded before being presented to the instance.

Retrieving the Data

An instance retrieves the data by querying a web server using a REST-like API. The base URI of all requests is `http://169.254.169.254/1.0/` where `1.0` indicates the API version.

The latest version of the API is always available using the URI `http://169.254.169.254/latest`.

Security of Launch Data

Although this data is only accessible by your specific instance, the data is not protected by cryptographic methods. You should take suitable precautions to protect sensitive data (such as long lived encryption keys).

You are not billed for these HTTP requests.

Retrieving Metadata

Requests for a specific metadatum resource returns the appropriate value or a `404` HTTP error code if the resource is not available. All metadata is returned as text (content type `text/plain`).

Requests for a general metadatum resource (i.e. an URI ending with a `/`) return a list of the resources available at that level or a `404` HTTP error code if there is no such resource. The list items are on separate lines with lines terminated by any combination of linefeed (ASCII 10) and carriage return (ASCII 13).

Resource	URI	Example

Get the available API versions	`GET http://169.254.169.254/`	Request
		
		`GET http://169.254.169.254/`
		
		Response
		
		`1.0`
		`2.0`

Get the top-level metadata items	`GET http://169.254.169.254/1.0/meta-data/`	Request
		
		`GET http://169.254.169.254/1.0/meta-data/`
		
		Response
		
		`ami-id`
		`ami-launch-index`
		`ami-manifest-path`
		`instance-id`
		`hostname`
		`local-ipv4`
		`public-keys/`
		`reservation-id`
		`security-groups`

Get the value of metadatum X (where 'X' is from the above list)	`GET http://169.254.169.254/1.0/meta-data/X`	Request
		
		`GET http://169.254.169.254/1.0/meta-data/ami-manifest-path`
		
		Response
		
		`my-amis/`

Resource	URI	Example

		`spamd-image.manifest.xml`
		
		Request
		
		`GET ht-tp://169.254.169.254/1.0/meta-data/ami-id`
		
		Response
		
		`ami-5bae4b32`
		
		Request
		
		`GET ht-tp://169.254.169.254/1.0/meta-data/reservation-id`
		
		Response
		
		`r-fea54097`
		
		Request
		
		`GET ht-tp://169.254.169.254/1.0/meta-data/hostname`
		
		Response
		
		`domU-12-34-31-00-00-05.usma1.compute.amazonaws.com`

Get the list of available public keys	`GET ht-tp://169.254.169.254/1.0/meta-data/public-keys/`	Request
		
		`GET ht-tp://169.254.169.254/1.0/meta-data/public-keys/`
		
		Response
		
		`0=my-public-key`

In which formats is public key 0 available?	`GET ht-tp://169.254.169.254/1.0/meta-data/public-keys/0/`	Request
		
		`GET ht-tp://169.254.169.254/1.0/meta-data/public-keys/0/`
		
		Response
		
		`openssh-key`

Get public key 0 (in openssh-key format)	`GET ht-tp://169.254.169.254/1.0/meta-data/pub-`	Request `GET ht-tp://169.254.169.254/1.0/meta-`

Resource	URI	Example

	`lic-keys/0/openssh-key`	`data/pub-lic-keys/0/openssh-key`
		
		Response
		
		`ssh-rsa AAAA.....wZEf my-public-key`

Retrieving User Data

Requests for the user data returns the data as-is (content type `application/x-octetstream`).

> **Note**
> As mentioned previously, all user-supplied data is treated as opaque data: what you give us is what you get back. It is thus the responsibility of the instance to interpret this data appropriately.

Resource	URI	Examples

Get the user-supplied data	`GET ht-tp://169.254.169.254/1.0/user-data`	Request
		
		`GET ht-tp://169.254.169.254/1.0/user-data`
		
		Response
		
		`1234,fred,reboot,true | 4512,jimbo, | 173,,,`
		
		Request
		
		`GET ht-tp://169.254.169.254/1.0/user-data`
		
		Response
		
		`[general]`
		`instances: 4`
		
		`[instance-0]`
		`s3-bucket: fred`
		
		`[instance-1]`
		`reboot-on-error: yes`
		
		Request
		
		`GET ht-tp://169.254.169.254/1.0/user-data`
		
		Response
		
		`GIF89aXfgs13qa....`

Example of Using the AMI Launch Index Value

Alice wants four instances of her favorite database AMI. The first instance will be the master with the remainder acting as replicants.

The master database configuration specifies various database parameters (the size of store, say) while the replicants' configuration specifies different parameters (replication strategy say). Alice decides to provide this data as an ASCII string with | delimiting the various instances' data:

```
store-size=123PB backup-every=5min | replicate-every=1min | replicate-
every=2min | replicate-every=10min | replicate-every=20min
```

The example above breaks down as follows

- `store-size=123PB backup-every=5min` defines the master database configuration
- `replicate-every=1min` defines the first replicant's configuration
- Etc.

Alice launches her instances:

```
$ ec2-run-instances ami-5bae4b32 -n 4 -d "store-size=123PB backup-every=5min
| replicate-every=1min | replicate-every=2min | replicate-every=10min | rep-
licate-every=20min"
RESERVATION     r-fea54097      598916040194    default
INSTANCE i-3ea74257 ami-5bae4b32 pending 0
INSTANCE i-31a74258 ami-5bae4b32 pending 1
INSTANCE i-31a74259 ami-5bae4b32 pending 2
INSTANCE i-31a7425a ami-5bae4b32 pending 3
```

Note that only 4 instances were launched.

Once launched, the instances all have a copy of the user data and the common metadata:

- AMI id: ami-5bae4b32
- AMI manifest path: ec2-public-images/getting-started.manifest.xml
- Reservation id: r-fea54097
- Public keys: none
- Security group names: default

However each instance has certain unique metadata:

Metadatum	Value
instance-id	i-3ea74257
ami-launch-index	0
hostname	domU-12-43-33-00-01-27.usma1.compute.amazonaws.com
local-ipv4	216.182.228.87

Metadatum	Value
instance-id	i-31a74258

Metadatum	Value
ami-launch-index	1
hostname	domU-12-31-33-00-01-72.usma1.compute.amazonaws.com
local-ipv4	216.182.228.88

Metadatum	Value
instance-id	i-31a74259
ami-launch-index	2
hostname	domU-12-31-33-00-01-73.usma1.compute.amazonaws.com
local-ipv4	216.182.228.89

Metadatum	Value
instance-id	i-31a7425a
ami-launch-index	3
hostname	domU-12-31-33-00-01-74.usma1.compute.amazonaws.com
local-ipv4	216.182.228.90

Therefore an instance can determine its portion of the user-supplied data by the simple process of

1. Determining which instance in the launch group it is:

```
GET http://169.254.169.254/1.0/meta-data/ami-launch-index
          1
```

2. Retrieving the user data:

```
GET http://169.254.169.254/1.0/user-data
          store-size=123PB backup-every=5min | replicate-every=1min | rep-
licate-every=2min | replicate-every=10min | replicate-every=20min
```

3. Extracting the appropriate part of the user data:

```
user_data.split('|')[ami_launch_index]
```

Using Shared AMIs

Introduction

This section looks at how to find and safely use shared AMIs.

Finding Shared AMIs

The following command displays a list of all public AMIs.

```
PROMPT> ec2dim -x all
```

The `-x all` flag shows AMIs executable by all users. This includes AMIs you own.

To show AMIs for which you have explicit launch permissions, run:

```
PROMPT> ec2dim -x self
```

The `-x self` flag shows AMIs you have explicit launch permissions for. AMIs you own are excluded.

To show AMIs owned by Amazon run:

```
PROMPT> ec2dim -o amazon
```

To find AMIs owned by a particular user run:

```
PROMPT> ec2dim -o 495219933132
```

Replace `495219933132` with the AWS account id of the user who owns the AMIs you are looking for.

Safely Using Shared AMIs

AMIs are launched at the user's own risk. Amazon cannot vouch for the integrity or security of AMIs shared by other users. Therefore, you should treat shared AMIs as you would any foreign code that you might consider deploying in your own data center and perform the appropriate due diligence.

Ideally, you will get the AMI ID from a trusted source (a website, another user, etc). If you do not know the source of an AMI, we recommended that you at leaste search the forums for comments on the AMI before launching it. Conversely, if you have questions or observations about a shared AMI, feel free to use the forums to ask or comment.

Amazon's public images have an aliased owner and will display `amazon` in the userId field. This allows users to find Amazon's public images easily.

Note

Users are not currently able to alias an AMI's owner.

If you do choose to launch a shared AMI, there are a number of steps you should take (at a minimum) after launch to confirm the AMI is not doing anything malicious:

- Check the ssh authorized keys file. The only key in the file should be the key you launched the AMI with.
- Check open ports and running services.
- Change the root password if is is not randomized on startup. Take a look at the section called “Disable Password Based Logins for Root” for more information on randomizing the root password on startup.

- Check if ssh allows root password logins. the section called “Disable Password Based Logins for Root” contains more information on disabling root based password logins.
- Check if there are any other user accounts that may allow backdoor entry to your instance. Accounts with super user privileges are paticularly dangerous.
- Check that all cron jobs are legitimate.

Using Get Console Output and Reboot Instances

Introduction

Amazon EC2 instances don't have a physical monitor to display their console output on. They also don't have physical controls to allow them to be powered-up, rebooted or shutdown. Instead these actions are enabled via the EC2 SOAP and Query APIs.

Console output is a valuable tool for problem diagnosis. It is especially useful for troubleshooting kernel problems and service configuration issues that may cause an instance to terminate or become unreachable before its ssh daemon can be started. Amazon EC2 provides a way to programmatically access instance console output via both the SOAP and Query APIs and the corresponding command-line tool.

Similarly, the ability to reboot instances that are otherwise unreachable is valuable for both trouble-shooting and general instance management. Amazon EC2 provides such a facility via the SOAP and Query APIs and the corresponding command-line tool.

Get Console Output

Amazon EC2 instance console output reflects exactly the character based console output that would otherwise be displayed on a physical monitor attached to a machine. This output is buffered as it is produced by the instance and then posted to a store from which it can be retrieved by the instance's owner. The posted output is not continuously updated. Rather, it is updated shortly after instance boot, reboot and once the instance terminates when it is likely to be of most value. Only the most recent 64KB of posted output is stored and it is available for a period of at least 1 hour after the last posting.

The console output for an instance can be retrieved via the SOAP API call described in the section called “GetConsoleOutput” and the Query API call described in the section called “GetConsoleOutput”. The corresponding command line tool, described in the section called “ec2-get-console-output”, can be used to retrieve the console output for an instance and display it to the user.

Console output can only be accessed by the instance owner.

Reboot Instances

As machines can be rebooted by pressing the reset button, EC2 instances can be rebooted via the SOAP API described in the section called “RebootInstances” and the Query API described in the section called “RebootInstances”. The corresponding command line tool described in the section called “ec2-reboot-instances” can be used to reboot a set of specified instances from the command-line.

Securing the Network

The Amazon EC2 service provides the ability to dynamically add and removed instances. However, this flexibility can complicate firewall configuration and maintenance which traditionally relies on IP addresses, subnet ranges or DNS host names as the basis for the firewall rules.

The Amazon EC2 firewall allows you to assign your compute resources to user-defined groups and define firewall rules for and in terms of these groups. As compute resources are added to or removed from groups, the appropriate rules are enforced. Similarly, if a group's rules are changed these changes are automatically applied to all members of the affected group.

Notes

- Defining firewall rules in terms of groups is flexible enough to allow you to implement functionality equivalent to a VLAN.
- In addition to the distributed firewall, you can maintain your own firewall on any of your instances. This may be useful if you have specific requirements not catered for by the distributed firewall.

Anticipated API changes

At present, the API calls for authorizing and revoking permissions are still under development. The remainder of this section outlines what you can depend on from this part of our API. The command line API tools expose only the subset of the functionality that is expected to remain unchanged.

Callers may depend on, now and in future, being able to grant permissions to

- source address ranges (specified with CIDRs, specific protocol and ports (or ICMP type/code)).
- source {user,group} tuples. No additional granularity, such as protocol and port (or ICMP type/code), should be expected.

Concepts

Security Groups

A security group is a named collection of access rules. These access rules specify which ingress, i.e. incoming, network traffic should be delivered to your instance. All other ingress traffic will be discarded.

A group's rules may be modified at any time. The new rules are automatically enforced for all running, as well as for subsequently launched, instances affected by the change in rules.

Note: Currently there is a limit of one hundred rules per group.

Group Membership

When an AMI instance is launched it may be assigned membership to any number of groups.

If no groups are specified, the instance is assigned to the "default" group. This group can be modified, by you, like any other group you have created. Be default, this group allows all network traffic from other members of the "default" group and discards traffic from other IP addresses and groups.

Group Access Rights

The access rules define source based access either for named security groups or for IP addresses, i.e. CIDRs. For CIDRs you may also specify the protocol and port range (or ICMP type/code).

Examples

We illustrate the use of the Amazon EC2 firewall in the following two examples. Note that we use the command line tools throughout the examples. The same results can be achieved using the SOAP API.

Default Group

1. Albert launches a copy of his favourite public AMI

   ```
   $ ec2-run-instances ami-eca54085
   RESERVATION     r-01927768      598916040194
   INSTANCE        i-cfd732a6      ami-eca54085            pending  0
   ```

2. After a little wait for image launch to complete, Albert, who is a cautious type, checks the access rules of the default group

   ```
   $ ec2-describe-group default
   GROUP   598916040194    default default group
   PERMISSION      default  ALLOWS  all                     FROM    USER
   598916040194    GRPNAME default
   ```

   and notices that it only accepts ingress network connections from other members of the default group for all protocols and ports.

3. Albert, being paranoid as well as cautious, port scans his instance

   ```
   $ nmap -P0 -p1-100 domU-12-31-33-00-01-56.usma1.compute.amazonaws.com
   Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 15:42
   SAST
   All 100 scanned ports on domU-12-31-33-00-01-56.usma1.compute.amazonaws.com
   (216.182.228.116) are: filtered

   Nmap finished: 1 IP address (1 host up) scanned in 31.008 seconds
   ```

4. Albert decides he should be able to SSH into his instance, but only from his own machine

   ```
   $ ec2-authorize default -P tcp -p 22 -s 192.168.1.130/32
   GROUP           default
   PERMISSION              default ALLOWS  tcp     22      22      FROM
   CIDR    192.168.1.130/32
   ```

5. Repeating the port scan

   ```
   $ nmap -P0 -p1-100 domU-12-31-33-00-01-56.usma1.compute.amazonaws.com
   Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 15:43
   SAST
   Interesting ports on domU-12-31-33-00-01-56.usma1.compute.amazonaws.com
   (216.182.228.116):
   (The 99 ports scanned but not shown below are in state: filtered)
   PORT   STATE SERVICE
   22/tcp open  ssh

   Nmap finished: 1 IP address (1 host up) scanned in 32.705 seconds
   ```

   Albert is happy (or at least less paranoid).

Three Tier Web Service

Mary wishes to deploy her public, fault tolerant, three tier web service in Amazon EC2. Her grand plan is to have her web tier start off executing in seven instances of ami-fba54092, her application tier executing in twenty instances of ami-e3a5408a, and her multi-master database in two instances of

ami-f1a54098. She's concerned that nasty people might gain access to her subscriber database, so she wants to restrict network access to her middle and back tier machines. When the traffic to her site increases over the holiday shopping period, she adds additional instances to her web and application tiers to handle the extra load.

1. First she creates a group for her Apache web server instances and allows HTTP access to the world

   ```
   $ ec2-add-group apache -d "Mary's Apache group"
   GROUP   apache  Mary's Apache group

   $ ec2-describe-group apache
   GROUP   598916040194    apache  Mary's Apache group

   $ ec2-authorize apache -P tcp -p 80 -s 0.0.0.0/0
   GROUP           apache
   PERMISSION              apache  ALLOWS  tcp     80      80      FROM
   CIDR    0.0.0.0/0

   $ ec2-describe-group apache
   GROUP   598916040194    apache  Mary's Apache group
   PERMISSION      598916040194    apache  ALLOWS  tcp     80      80
   FROM    CIDR    0.0.0.0/0
   ```

   She then launches seven instances of her web server AMI as members of this group

   ```
   $ ec2run ami-fba54092 -n 7 -g apache
   RESERVATION     r-01927768      598916040194
   INSTANCE        i-cfd732a6      ami-fba54092            pending
   ...

   $ ec2din i-cfd732a6
   RESERVATION     r-0592776c      598916040194
   INSTANCE        i-cfd732a6      ami-fba54092       domU-
   12-31-33-00-04-16.usma1.compute.amazonaws.com      running
   ...
   ```

   Having studied at the same school of paranoia as Albert, Mary does a port scan to confirm the permissions she just configured

   ```
   $ nmap -P0 -p1-100 domU-12-31-33-00-04-16.usma1.compute.amazonaws.com
   Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 16:21
   SAST
   Interesting ports on domU-12-31-33-00-04-16.usma1.compute.amazonaws.com
   (216.182.231.20):
   (The 99 ports scanned but not shown below are in state: filtered)
   PORT   STATE SERVICE
   80/tcp open  http

   Nmap finished: 1 IP address (1 host up) scanned in 33.409 seconds
   ```

   And then she tests to make sure her web server is contactable

   ```
   $ telnet domU-12-31-33-00-04-16.usma1.compute.amazonaws.com 80
   Trying 216.182.231.20...
   Connected to domU-12-31-33-00-04-16.usma1.compute.amazonaws.com
   (216.182.231.20).
   Escape character is '^]'.
   ```

   Excellent!

2. She now creates a separate group for her application server

   ```
   $ ec2-add-group appserver -d "Mary's app server"
   GROUP   appserver       Mary's app server
   ```

then starts twenty instances as members of this group

```
$ ec2run ami-e3a5408a -n 20 -g appserver
```

and grants network access between her web server group and the application server group

```
$ ec2-authorize appserver -o apache -u 598916040194
GROUP           appserver
PERMISSION      appserver  ALLOWS  all                     FROM    USER
598916040194    GRPNAME apache
```

She checks to ensure access to her app server is indeed restricted by port scanning one of the app servers

```
$ nmap -P0 -p1-100 domU-12-31-33-00-03-D1.usma1.compute.amazonaws.com
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2006-08-07 15:42
SAST
All 100 scanned ports on domU-12-31-33-00-03-D1.usma1.compute.amazonaws.com
(216.182.228.12) are: filtered

Nmap finished: 1 IP address (1 host up) scanned in 31.008 seconds
```

3. To confirm that her web servers have access to her application servers she needs to do a little extra work...
   a. She (temporarily) grants SSH access from her workstation to the web server group

      ```
      $ ec2-authorize apache -P tcp -p 22 -s 192.168.1.130/32
      ```

   b. She logs in to one of her web servers and connects to an application server on TCP port 8080

      ```
      $ telnet domU-12-31-33-00-03-D1.usma1.compute.amazonaws.com 8080
      Trying 216.182.228.12...
      Connected to domU-12-31-33-00-03-D1 .usma1.compute.amazonaws.com
      (216.182.228.12).
      Escape character is '^]'
      ```

   c. Satisfied with the setup, she revokes SSH access to the web server group

      ```
      $ ec2-revoke apache -P tcp -p 22 -s 192.168.1.130/32
      ```

Creating the group for database servers and granting access to them from the application server group is left as an exercise for the reader ;-)

Tools and APIs

Below we highlight the most relevant command-line tools and SOAP API calls used to manipulate security groups. Please refer to the appropriate sections of this guide for the specific details.

Purpose	Command-line tool	SOAP API
List the rules belonging to specified groups	`ec2-describe-group`	DescribeSecurityGroups
Create a new security group	`ec2-add-group`	CreateSecurityGroup
Delete an existing security group	`ec2-delete-group`	DeleteSecurityGroup
Add an access rule to an existing security group	`ec2-authorize`	AuthorizeSecurityGroupIngress
Remove an access rule from an existing security group	`ec2-revoke`	RevokeSecurityGroupIngress

Using the APIs

This section details the APIs available. Currently the APIs are available as SOAP calls and HTTP Query requests.

Using the SOAP API

WSDL and Schema Definitions

The Amazon EC2 web service can be accessed using the SOAP web services messaging protocol. This interface is described by a Web Services Description Language (WSDL) document which defines the operations and security model for the service. The WSDL references an XML Schema document which strictly defines the datatypes that may appear in SOAP requests and responses. For more information on WSDL and SOAP, please see the references in the section called “Additional Web Services References”.

All schemas have a version number. The version number appears in the URL of a schema file, and in a schema's target namespace. The latest version is 2007-01-03. Upgrading is made easy by differentiating requests based on the version number. In addition to the latest version, the service will support the older versions for some time. Once customer transition to the new version is complete, the older versions will be retired.

The Amazon EC2 services API WSDL can be found at URLs of the form 'http://ec2.amazonaws.com/doc/VERSION/ec2.wsdl' where VERSION indicates the version of the API. The current API version is 2007-01-03 and can thus be found at URL http://ec2.amazonaws.com/doc/2007-01-03/AmazonEC2.wsdl

Making Requests

The Amazon EC2 web service complies with the current WS-Security standard, requiring SOAP request messages to be hashed and signed for integrity and non-repudiation. WS-Security defines profiles which are used to implement various levels of security. Amazon EC2 secure SOAP messages use BinarySecurityToken profile, consisting of an X.509 certificate with an RSA public key.

Programming Language Support in Amazon EC2

Since the SOAP requests and responses in the Amazon EC2 Web Service follow current standards, any programming language with the appropriate library support may be used. Languages known to have such support include C++, C#, Java, Perl, Python and Ruby. Currently we only supply java libraries for our API but expect to release additional language bindings in the future.

Request Authentication

The following is an insecure request to run instances:

```
<RunInstances xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
    <instancesSet>
        <item>
            <imageId>ami-60a54009</imageId>
            <minCount>1</minCount>
            <maxCount>3</maxCount>
        </item>
    </instancesSet>
    <groupSet/>
</RunInstances>
```

In order to secure the request, we must add the BinarySecurityToken element mentioned above. The Java libraries we supply rely on the Apache Axis project for XML security, canonicalization and SOAP support. (The Sun Java Web Service Developer's Pack supplies libraries of equivalent functionality.)

The secure version of the request begins with the following:

```
<SOAP-ENV:Envelope xm-
lns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security xm-
lns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
      <wsse:BinarySecurityToken
      xm-
lns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-u
tility-1.0.xsd"
      Encoding-
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-se
curity-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token
-profile-1.0#X509v3"
      wsu:Id="CertId-1064304">....many, many lines of base64 encoded
      X.509 certificate...</wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Al-
gorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:CanonicalizationMethod
>
          <ds:SignatureMethod Al-
gorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></ds:SignatureMethod>
          <ds:Reference URI="#id-17984263">
            <ds:Transforms>
              <ds:Transform Al-
gorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Al-
gorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
            <ds:DigestValue>0pjZ1+TvgPf6uG7o+Yp3l2YdGZ4=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#id-15778003">
            <ds:Transforms>
              <ds:Transform Al-
gorithm="http://www.w3.org/2001/10/xml-exc-c14n#"></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Al-
gorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
            <ds:DigestValue>HhRbxBBmc2OO348f8nLNZyo4AOM=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
<ds:SignatureValue>bmVx24Qom4kd9QQtclxWIlgLk4QsQBPaKESi79x479xgbO9PEStXMiHZuB
Ai9luuKdNTcfQ8UE/d
        jjHKZKEQR-
COlLVy0Dn5ZL1RlMHsv+OzJzzvIJFTq3LQKNrzJzsNe</ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-17007273">
          <wsse:SecurityTokenReference
             xm-
lns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-u
tility-1.0.xsd" wsu:Id="STRId-22438818">
            <wsse:Reference URI="#CertId-1064304"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token
-profile-1.0#X509v3">
            </wsse:Reference>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
      <wsu:Timestamp
         xm-
lns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-u
tility-1.0.xsd" wsu:Id="id-17984263">
        <wsu:Created>2006-06-09T10:57:35Z</wsu:Created>
        <wsu:Expires>2006-06-09T11:02:35Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </SOAP-ENV:Header>
```

Let's take a quick look at the most important elements in case you are matching this against requests generated by Amazon EC2 supplied libraries, or those of another vendor.

- BinarySecurityToken - contains the X.509 certificate in base64 encoded PEM format.
- Signature - contains XML digital signature created using the canonicalization, signature algorithm, and digest method described within.
- Timestamp - Any request is only valid to Amazon EC2 within 5 minutes of this value. Used to prevent replay attacks.

Understanding Responses

In response to a request, the Amazon EC2 web service returns an XML data structure that conforms to an XML schema defined as part of the Amazon EC2 WSDL. The structure of a XML response is specific to the associated request. In general, the response datatypes with be named according to the operation performed and whether the datatype is a container (may have children). Examples of containers include 'groupSet' for security groups and 'instancesSet' for instances. Item elements are children of containers and their contents vary according to the container's role.

An example response is:

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
    <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
    </item>
    <item>
      <instanceId>i-2bc64242</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
    <name>pending</name>
      </instanceState>
      <dnsName>domU-13-35-33-00-00-5C.dc2.compute.amazonaws.com</dnsName>
    </item>
    <item>
      <instanceId>i-2be64332</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
    <name>pending</name>
      </instanceState>
      <dnsName>domU-12-34-28-00-00-5C.dc2.compute.amazonaws.com</dnsName>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

Additional Web Services References

- Web Service Description Language (WSDL)
- WS-Security BinarySecurityToken Profile

Using the Query API

Making Requests

HTTP Query-based requests are defined as any HTTP requests using the HTTP verb GET or POST and a Query parameter named either Action or Operation. Action is used throughout this documentation, although Operation is supported for backward compatibility with other AWS Query APIs.

Query Parameters

Each Query request must include some common parameters to handle authentication and selection of an action. These parameters are documented in the section called “Common Query Parameters”.

Some operations take lists of parameters. These lists are specified using the *param.n* notation. Values of *n* should be integers starting from 1.

Query API Authentication

Every request to Amazon EC2 must contain a request signature. A request signature is calculated by constructing a string and then calculating an RFC 2104-compliant HMAC-SHA1 hash, using the Secret AWS Access Key as the key. For more information, see http://-www.faqs.org/-rfcs/-rfc2104.html.

The following are the basic steps used in authenticating requests to AWS. It is assumed that the developer has already registered with AWS and received an Access Key ID and Secret Access Key.

1. The sender constructs a request to AWS.
2. The sender calculates the request signature, a Keyed-Hashing for Message Authentication Code (HMAC) with a SHA-1 hash function, as defined in the next section of this topic.
3. The sender of the request sends the request data, the signature, and Access Key ID (the key-identifier of the Secret Access Key used) to AWS.
4. AWS uses the Access Key ID to look up the Secret Access Key.
5. AWS generates a signature from the request data and the Secret Access Key using the same algorithm used to calculate the signature in the request.
6. If the signatures match, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response.

Note
If a request contains a Timestamp parameter, the signature calculated for the request expires 15 minutes after the Timestamp value. If a request contains an Expires parameter, the signature expires at the time specified as the value for the Expires parameter.

Calculating Request Signatures

The following steps demonstrate how to calculate a signature for requests to AWS:

1. Based on the API (Query/SOAP/REST) being used, construct a string.
2. Compute an RFC 2104 compliant HMAC using the Secret AWS Access Key as the "key". This value should be base64 encoded, and then included as the value for the Signature parameter for the request.

Calculating the string to sign

The following steps demonstrate how to calculate the string to be signed:

1. The query parameters (not URL-encoded) need to be sorted case-insensitively.
2. Concatenate the parameter names and values without the initial `?` or the separating `&` and `=` characters.

Given the following Query string to sign (linebreaks added for clarity):

```
?Action=DescribeImages
&AWSAccessKeyId=10QMXFEV71ZS32XQFTR2
&SignatureVersion=1
&Timestamp=2006-12-08T07%3A48%3A03Z
&Version=2007-01-03
```

The HMAC signature should be calculated over the following string:

```
ActionDescribeImagesAWSAccessKey-
Id10QMXFEV71ZS32XQFTR2SignatureVersion1Timestamp2006-12-08T07:48:03ZVersion20
07-01-03
```

Calculating the HMAC signature

Given the Query string above and the secret key `DMADSSfPfdaDjbK+RRUhS/aDrjsiZadgAUm8gRU2` the base64 encoded signature is as follows:

```
GjH3941IBe6qsgQu+k7FpCJjpnc=
```

Shown below is a Java code sample to compute the signature from the string and the private key.

```
import java.security.SignatureException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class HmacExample
{

    private static final String HMAC_SHA1_ALGORITHM = "HmacSHA1";

    /**
     * Computes RFC 2104-compliant HMAC signature.
     *
     * @param data
     *     The data to be signed.
     * @param key
     *     The signing key.
     * @return
     *     The base64-encoded RFC 2104-compliant HMAC signature.
     * @throws
     *     java.security.SignatureException when signature generation fails
     */
    public static String calculateRFC2104HMAC(String data, String key)
        throws java.security.SignatureException
    {
        String result;
        try {
            // get an hmac_sha1 key from the raw key bytes
            SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(),
HMAC_SHA1_ALGORITHM);

            // get an hmac_sha1 Mac instance and initialize with the signing
key
```

```
            Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
            mac.init(signingKey);

            // compute the hmac on input data bytes
            byte[] rawHmac = mac.doFinal(data.getBytes());

            // base64-encode the hmac
            result = Base64.encodeBytes(rawData);
        }
        catch (Exception e) {
            throw new SignatureException("Failed to generate HMAC : " +
e.getMessage());
        }
        return result;
    }
}
```

Example Request

Here is a complete example request, including all required parameters:

```
?AWSAccessKeyId=10QMXFEV71ZS32XQFTR2&Action=DescribeImages&SignatureVersion=1
&Timestamp=2006-12-08T07%3A48%3A03Z&Version=2007-01-03&Signature=GjH3941IBe6q
sgQu%2Bk7FpCJjpnc%3D
```

API Reference

Amazon EC2 provides two APIs, SOAP and Query. These APIs allow developers to launch and control instances from their own applications.

This section discusses the operations available in the Amazon EC2 APIs, the semantics of those calls and the parameters that must be supplied. Examples of requests and responses are also provided.

Note
The same XML body is returned in both the Query API and SOAP API.

We recommend you familiarize yourself with the conventions we've used in describing the API.

API Conventions

Overview

This topic discusses the conventions used in the Amazon EC2 API reference. This includes terminology, notation and any abbreviations used to illuminate the API.

The API reference is broken down into a collection of *Actions* and *Data Types*.

Actions

Actions encapsulate the possible interactions with Amazon EC2. These can be viewed as remote procedure calls and consist of a request and response message pair. Requests must be signed, allowing Amazon EC2 to authenticate the caller. For clarity, the sample requests and responses illustrating each of the operations described in this reference are not signed.

Data Types and the Amazon EC2 WSDL

The current version of the Amazon EC2 WSDL is available at the following location: http://ec2.amazonaws.com/doc/2007-01-03/AmazonEC2.wsdl. Some libraries can generate code directly from the WSDL. Other libraries require a little more work on your part.

Values provided as parameters to the various operations must be of the indicated type. Standard XSD types (like `string`, `boolean`, `int`) are prefixed with `xsd:`. Complex types defined by the Amazon EC2 WSDL are prefixed with `ec2:`.

Parameters that consist of lists of information are defined within our WSDL to require <info> tags around each member. Throughout the API, type references for parameters that accept such a list of values are specified using the notation `type[]` The type referred to in these instances is the type *nested within the <info> tag* (for Amazon EC2 types this is defined in the WSDL).

For example, the <imagesSet> element in the following XML snippet is of type `xsd:string[]`:

```
<imagesSet>
  <item>
    <imageId>ami-61a54008</imageId>
  </item>
  <item>
    <imageId>ami-61b54608</imageId>
  </item>
</imagesSet>
```

And the <instancesSet> element in the following XML snippet is of type `ec2:RunInstanceItemType[]`:

```
<instancesSet>
    <item>
        <imageId>ami-60a54009</imageId>
        <minCount>10</minCount>
        <maxCount>30</maxCount>
    </item>
    <item>
        <imageId>ami-60b54209</imageId>
        <minCount>5</minCount>
        <maxCount>20</maxCount>
    </item>
</instancesSet>
```

API Versioning

All Amazon EC2 API updates are versioned. This helps to minimize the impact of API changes on client software by making it possible to always send back a response that the client is capable of processing. We endeavor as far as possible to retain backwards compatibility with new API revisions. However, there may be occasions where an incompatible API change is required. In addition, in newer API releases existing responses may include additional fields, and depending on how client software is written it may or may not be able to handle these additional fields. By including a version in the request, a client guarantees that it will always be sent a response it expects.

Each API revision is assigned a version in date form (the current API version is `2007-01-03`). This version is included in the request as part of the document namespace when using our SOAP API and as a `Version` parameter when using our Query API. The response returned by Amazon EC2 will honor the version included in the request. Fields introduced in a later API version will not be returned in the response.

SOAP clients that retrieve the Amazon EC2 WSDL at runtime and generate their requests dynamically using that WSDL should reference the WSDL for the version of the API the client was developed against. This will ensure client software continues to work even in the face of backwards incompatible API changes. The WSDL for each supported API version is available from the following URI:

```
http://ec2.amazonaws.com/doc/<api-version>/AmazonEC2.wsdl
```

The WSDL for latest version of our API can always be retrieved from the following URI:

```
http://ec2.amazonaws.com/doc/AmazonEC2.wsdl
```

Note

The WSDL referenced in the above link should be treated as a moving target. This WSDL will always track the latest release of the Amazon EC2 SOAP API. If your software depends on fetching the WSDL at runtime then we strongly recommend you reference the specific version of the WSDL you are developing against.

API Error Codes

There are two types of error codes, client and server.

Client error codes suggest that the error was caused by something the client did, such as an authentication failure or an invalid AMI identifier. In the SOAP API, These error codes are prefixed with `Client.` For example: `Client.AuthFailure`. In the Query API, these errors are accompanied by a `40x` HTTP response code.

Server error codes suggest that the error was caused by a server-side issue, and should be reported. In the SOAP API, These error codes are prefixed with `Server.` For example: `Server.Unavailable`. In the Query API, these errors are accompanied by a `50x` HTTP response code.

Summary of Client Error Codes

Error Code	Definition	Notes
`Auth-Failure`	User not authorized.	Common cause is trying to run an AMI for which you do not have permission.
`Invalid-Manifest`	Specified AMI has an unparsable Manifest.	
`Invalid-AMIID.Malformed`	Specified AMI ID is not valid.	
`Invalid-AMIID.NotFound`	Specified AMI ID does not exist.	
`Invalid-AMIID.Unavailable`	Specified AMI ID has been deregistered and is no longer available.	
`Invalid-InstanceID.Malformed`	Specified instance ID is not valid.	
`Invalid-InstanceID.NotFound`	Specified instance ID does not exist.	
`InvalidKeyPair.NotFound`	Specified keypair name does not exist.	
`InvalidKeyPair.Duplicate`	Attempt to create a duplicate keypair.	

Error Code	Definition	Notes
`Invalid-Group.NotFound`	Specified group name does not exist.	
`Invalid-Group.Duplicate`	Attempt to create a duplicate group.	
`Invalid-Group.InUse`	Specified group can not be deleted because it is in use.	
`Invalid-Group.Reserved`	Specified group name is a reserved name.	
`Invalid-Parameter-Value`	The value supplied for a parameter was invalid.	Requests that could cause this error include (for example) supplying an invalid image attribute to the `DescribeImageAttribute` request or an invalid `version` or `encoding` value for the `user-Data` in a `RunInstances` request.
`Invalid-Permission.Duplicate`	Attempt to authorize a permission that has already been authorized.	
`Invalid-Permission.Malformed`	Specified permission is invalid.	
`InvalidReservationID.Malformed`	Specified reservation ID is invalid.	
`InvalidReservationID.NotFound`	Specified reservation ID does not exist.	
`InstanceLimitExceeded`	User has max allowed concurrent running instances.	Each user has a concurrent running instance limit. For new users during public beta, this limit is 20.
`Invalid-ParameterCombination`	RunInstances was called with minCount and maxCount set to 0 or minCount > maxCount.	
`Invalid-UserID.Malformed`	The user ID is niether in the form of an AWS account ID or one of the special values accepted by the `own-`	

Error Code	Definition	Notes
	`er` or `executableBy` flags in the DescribeImages call.	
`InvalidAMIAttributeItemValue`	The value of an item added to, or removed from, an image attribute is invalid.	If you are specifying a userId check that it is in the form of an AWS account ID.

Summary of Server Error Codes

Error Code	Definition	Notes
`InternalError`	Internal Error.	Should not occur. Please let us know. Try to reproduce.
`InsufficientInstanceCapacity`	Not enough available instances to satify your minimum request.	You can lower your request or wait for additional capacity to become available.
`Unavailable`	Indicates the server is overloaded and cannot handle request.	

Common Data Types

The Amazon EC2 API contains several data types used by the various operations. This section describes each operation in detail.

Since both the Query and SOAP APIs return the same XML body, the data types described in the WSDL are used in both.

DescribeImagesResponseItemType

The *DescribeImagesResponseItemType* data type.

Relevant Operations

Operations that use this data type include:

- DescribeImages

Contents

The following table describes and shows the elements contained in DescribeImagesResponseItemType.

Member	Description	Type
imageId	Unique ID of the AMI being described.	xsd:string
imageState	Current state of the AMI. • available: the image has been successfully registered and is available for launching • deregistered: the image has recently been deregistered and is no longer available for launching	xsd:string
imageOwnerId	AWS Access Key ID of the image owner.	xsd:string
isPublic	true if anyone other than the owner can launch instances of this image.	xsd:boolean

DescribeKeyPairsResponseItemType

The *DescribeKeyPairsResponseItemType* data type.

Relevant Operations

Operations that use this data type include:

- DeleteKeypair

- DescribeKeypairs

Contents

The following table describes and shows the elements contained in DescribeKeyPairsResponseItemType.

Member	Description	Type
keyName	The user supplied name for this key pair.	xsd:string
keyFingerprint	A fingerprint for the private key of this keypair. This is computed as the SHA-1 digest of the DER encoded form of the private key.	xsd:string

EmptyElementType

The *EmptyElementType* data type.

Relevant Operations

Operations that use this data type include:

- ResetImageAttribute
- DescribeImageAttribute

Contents

The empty element is just that - an empty element, and has no contents.

GroupSetType

The *GroupSetType* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances

Contents

The following table describes and shows the elements contained in GroupSetType.

Member	Description	Type
groupId	Name of a security group.	xsd:string

InstanceStateType

The *InstanceStateType* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances
- DescribeInstances
- TerminateInstances

Contents

The following table describes and shows the elements contained in InstanceStateType.

Member	Description	Type
code	A 16 bit unsigned integer. The high byte is an opaque internal value and should be ignored when consulting this value. The low byte is set based on the state represented: • `pending`: 0 • `running`: 16 • `shutting-down`: 32 • `terminated`: 48	`xsd:int`
name	The current state of the instance. • `pending`: the instance is in the process of being launched • `running`: the instance has been launched (although it may not yet have completed the boot process) • `shutting-down`: the instance has begun the shutdown process • `terminated`: the instance has been terminated	`xsd:string`

IpPermissionType

The *IpPermissionType* data type.

Relevant Operations

Operations that use this data type include:

- AuthorizeSecurityGroupIngress
- DescribeSecurityGroups
- RevokeSecurityGroupIngress

Contents

The following table describes and shows the elements contained in IpPermissionType.

Member	Description	Type
ipProtocol	IP Protocol.	`xsd:string`
fromPort	Start of port range for the TCP and UDP protocols, or an ICMP type number. An ICMP type number of -1 indicates a wildcard (i.e. any ICMP type number).	`xsd:int`
toPort	End of port range for the TCP and UDP protocols, or an ICMP code. An ICMP code of -1 indicates a wildcard (i.e. any ICMP code).	`xsd:int`
groups	List of security group and user ID pairs.	`ec2:UserIdGroupPairType[]`
ipRanges	List of CIDR IP range specifications.	`xsd:string[]`

LaunchPermissionItemType

The *LaunchPermissionItemType* data type.

Relevant Operations

Operations that use this data type include:

- ModifyImageAttribute
- DescribeImageAttribute

Contents

The following table describes and shows the elements contained in `LaunchPermissionItemType`.

Element Name	Description	Required?
group	A launch permission for a group. Currently only `all` is supported, which gives public launch permissions.	Choice between *group* and *userId*
userId	A launch permission for a user. *userId* is an AWS account id.	Choice between *group* and *userId*

LaunchPermissionOperationType

The *LaunchPermissionOperationType* data type.

Relevant Operations

Operations that use this data type include:

- ModifyImageAttribute

Contents

The following table describes and shows the elements contained in LaunchPermissionOperationType.

Element Name	Description	Required?
add	Adds launch permissions.	Choice between *add* and *remove*
remove	Removes launch permissions.	Choice between *add* and *remove*

ReservationInfoType

The *ReservationInfoType* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances
- DescribeInstances

Contents

The following table describes and shows the elements contained in ReservationInfoType.

Member	Description	Type
reservationId	Unique ID of the reservation being described.	xsd:string
ownerId	AWS Access Key ID of the user who owns the reservation.	xsd:string
groupSet	Set of security groups these instances were launched in.	ec2:GroupSetType[]
instancesSet	Information about instances started.	ec2:RunningInstancesItemType[]

RunInstanceItemType

The *RunInstanceItemType* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances

Contents

The following table describes and shows the elements contained in RunInstanceItemType.

Member	Description	Type
imageId	Unique ID of a machine image, returned by a call to RegisterImage.	xsd:string
minCount	Minimum number of instances to launch. If *minCount* is more than Amazon EC2 can launch, no instances are launched at all.	xsd:int
maxCount	Maximum number of instances to launch. If *maxCount* is more than Amazon EC2 can launch, the largest possible number above *minCount* will be launched instead.	xsd:int
keyName	The name of the keypair.	xsd:string

RunningInstancesItemType

The *RunningInstancesItemType* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances

Contents

The following table describes and shows the elements contained in RunningInstancesItemType.

Element Name	Description	Type
instanceId	Unique ID of the instance launched.	xsd:string
imageId	Image ID of the AMI used to launch the instance.	xsd:string
instanceState	The current state of the instance.	ec2:InstanceStateType

Element Name	Description	Type
	• `pending`: the instance is in the process of being launched • `running`: the instance has been launched (although it may not yet have completed the boot process) • `shutting-down`: the instance has begun the shutdown process • `terminated`: the instance has been terminated	
dnsName	The DNS name assigned to the instance. This element remains empty until the instance enters a running state.	`xsd:string`
reason	An optional reason for the most recent state transition. This may be an empty string.	`xsd:string`
keyName	An optional key name. If this instance was launched with an associated key pair, this is the name of that key pair.	`xsd:string`
amiLaunchIndex	An optional AMI launch index which can be used to determine which instance this is in the launch group. See using instance data for more info.	`xsd:string`

SecurityGroupItemType

The *SecurityGroupItemType* data type.

Relevant Operations

Operations that use this data type include:

- DescribeSecurityGroups

Contents

The following table describes and shows the elements contained in SecurityGroupItemType.

Member	Description	Type
ownerId	AWS Access Key ID of the owner of the security group described.	`xsd:string`
groupName	Name of the security group.	`xsd:string`
groupDescription	Description of the security group.	`xsd:string`

Member	Description	Type
ipPermissions	Set of IP permissions associated with the security group.	ec2:IpPermissionType[]

TerminateInstancesResponseInfoType

The *TerminateInstancesResponseInfoType* data type.

Relevant Operations

Operations that use this data type include:

- TerminateInstances

Contents

The following table describes and shows the elements contained in TerminateInstancesResponseInfoType.

Element Name	Description	Type
instanceId	Instance ID returned from previous call to RunInstances.	xsd:string

UserDataType

The *UserDataType* data type.

Relevant Operations

Operations that use this data type include:

- RunInstances

Contents

The following table describes and shows the elements contained in UserDataType.

Member	Description	Type
data	The user data.	xsd:string

Notes

- The *data* element must specify the attributes

Attribute name	Required?	Value
version	Yes	1.0
encoding	Yes	base64

- The user data is base64-encoded as per RFC3548 with the additional restrictions
  - Implementations MUST NOT add linefeeds to encoded data
  - Implementations MUST pad (end of) encoded data with '=' if required
  - Implementations MUST ignore characters in the encoded stream that are not in the encoding alphabet. Note that this differs from what RFC3548 says. It is included because it provides more leeway for clients.
  - Encoding alphabet as per table 1 in RFC3548 (i.e. A-Za-z0-9+/)
  - The size limit on the user data applies to the data before base64 encoding

UserIdGroupPairType

The *`UserIdGroupPairType`* data type.

Relevant Operations

Operations that use this data type include:

- AuthorizeSecurityGroupIngress
- DescribeSecurityGroups
- RevokeSecurityGroupIngress

Contents

The following table describes and shows the elements contained in UserIdGroupPairType.

Member	Description	Type
`userId`	AWS Access Key ID of a user.	`xsd:string`
`groupName`	Name of a security group.	`xsd:string`

EC2 SOAP API

The Amazon EC2 API consists of web service operations for every task the service can perform. This section describes each operation in detail.

By Function

Operations

Images

- RegisterImage
- DescribeImages
- DeregisterImage

Instances

- RunInstances
- DescribeInstances
- TerminateInstances

Keypairs

- CreateKeyPair
- DescribeKeyPairs
- DeleteKeyPair

Image Attributes

- ModifyImageAttribute
- DescribeImageAttribute
- ResetImageAttribute

Security Groups

- CreateSecurityGroup
- DescribeSecurityGroups
- DeleteSecurityGroup
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress

AuthorizeSecurityGroupIngress

The `AuthorizeSecurityGroupIngress` operation adds permissions to a security group.

Permissions are specified in terms of the IP protocol (TCP, UDP or ICMP), the source of the request (by IP range or an Amazon EC2 user-group pair), source and destination port ranges (for TCP and UDP), and ICMP codes and types (for ICMP).

Note
Changes are anticipated in this API that may restrict further what is allowable. Please consult the section called "Anticipated API changes" for more details.

Permission changes are propagated to instances within the security group being modified as quickly as possible. However, a small delay is likely, depending on the number of instances that are members of the indicated group.

Request Parameters

The following table describes the request parameters for `AuthorizeSecurityGroupIngress`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
userId	AWS Access Key ID.	Yes	`xsd:string`
groupName	Name of the group to modify.	Yes	`xsd:string`
ipPermissions	Set of permissions to add to the group.	Yes	`ec2:IpPermissionType[]`

Response Tags

The following table describes the default response tags included in `AuthorizeSecurityGroupIngress` responses.

Element Name	Definition
return	`true` if permissions successfully added.

Sample Request

```
<AuthorizeSecurityGroupIngress xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
    <userId/>
    <groupName>WebServers</groupName>
    <ipPermissions>
        <item>
            <ipProtocol>tcp</ipProtocol>
            <fromPort>80</fromPort>
            <toPort>80</toPort>
            <groups/>
            <ipRanges>
                <item>
                    <cidrIp>0.0.0.0/0</cidrIp>
                </item>
            </ipRanges>
        </item>
    </ipPermissions>
```

```
</AuthorizeSecurityGroupIngress>
```

Sample Response

```
<AuthorizeSecurityGroupIngressResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</AuthorizeSecurityGroupIngressResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

CreateKeyPair

The `CreateKeyPair` operation creates a new 2048 bit RSA keypair and returns a unique ID that can be used to reference this keypair when launching new instances.

Request Parameters

The following table describes the request parameters for `CreateKeyPair`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
keyName	A unique name for this key.	Yes	`xsd:string`

Response Tags

The following table describes the default response tags included in `CreateKeyPair` responses.

Element Name	Definition
keyName	The key name provided in the original request.
keyFingerprint	A SHA-1 digest of the DER encoded private key.
keyMaterial	An unencrypted PEM encoded RSA private key.

Sample Request

```
<CreateKeyPair xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <keyName>example-key-name</keyName>
</CreateKeyPair>
```

Sample Response

```
<CreateKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <keyName>example-key-name</keyName>
<keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f</
keyFingerprint>
  <keyMaterial>-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBuLFg5ujHrtm1jnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFxTHgElQiJLChp
HungXQ29VTc8rc1bW0lkdi23OH5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKCOVUR+jx71Sg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjGlUKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3LpGZyq/
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYGtGSBMpTRP5hnbzzuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAY1tsiUsIwDl5
91CXirkYGuVfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVD1pM
ZNUJs7rw9gZRTrf7LylaJ58kOcyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1kGQI7
3wcfgt5ecIu4TZf0OE9IHjn+2eRlsrjBdeORi7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhHOux5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLGCwU9h9HlO9mKAc2m8Cm1
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkpHi2EISh1uRkhxljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAy1/EDNTY5OSIJU+0KFmQbyhsbm
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfqGoQc7+sAc8vmKkesnF7CqCEKDyF/dhrxYdQKB
gC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeEO+uZ58/BGKOIGHByHBDiXtzMhdJr15HTYjxK7OgTZm
gK+8zp4L9IbvLGDMJO8vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+Zrvm1F0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0gij1houRe
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----</keyMaterial>
</CreateKeyPairResponse>
```

Related Operations

- DescribeKeyPairs
- DeleteKeyPair
- RunInstances

CreateSecurityGroup

The `CreateSecurityGroup` operation creates a new security group.

Every instance is launched in a security group. If none is specified as part of the launch request then instances are launched in the default security group. Instances within the same security group have unrestricted network access to one another. Instances will reject network access attempts from other instances in a different security group. As the owner of instances you may grant or revoke specific permissions using the AuthorizeSecurityGroupIngress and RevokeSecurityGroupIngress operations.

Request Parameters

The following table describes the request parameters for `CreateSecurityGroup`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
groupName	Name for the new security group.	Yes	`xsd:string`
groupDescrip-	Description of the new security group.	Yes	`xsd:string`

Element Name	Definition	Required?	Type
tion			

Response Tags

The following table describes the default response tags included in `CreateSecurityGroup` responses.

Element Name	Definition
return	`true` if call succeeded.

Sample Request

```
<CreateSecurityGroup xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
    <groupName>WebServers</groupName>
    <groupDescription>Web</groupDescription>
</CreateSecurityGroup>
```

Sample Response

```
<CreateSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</CreateSecurityGroupResponse>
```

Related Operations

- RunInstances
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

DeleteKeyPair

The `DeleteKeyPair` operation deletes a keypair.

Request Parameters

The following table describes the request parameters for `DeleteKeyPair`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
keyName	Name of the keypair to delete.	Yes	`xsd:string`

Response Tags

The following table describes the default response tags included in `DeleteKeyPair` responses.

Element Name	Definition
return	`true` if the key was successfully deleted.

Sample Request

```
<DeleteKeyPair xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
    <keyName>example-key-name</keyName>
</DeleteKeyPair>
```

Sample Response

```
<DeleteKeyPair xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</DeleteKeyPair>
```

Related Operations

- CreateKeyPair
- DescribeKeyPairs

DeleteSecurityGroup

The `DeleteSecurityGroup` operation deletes a security group.

If an attempt is made to delete a security group and any instances exist that are members of that group a fault is returned.

Request Parameters

The following table describes the request parameters for `DeleteSecurityGroup`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
groupName	Name of the security group to delete.	Yes	`xsd:string`

Response Tags

The following table describes the default response tags included in `DeleteSecurityGroup` responses.

Element Name	Definition
return	`true` if group deleted.

Sample Request

```
<DeleteSecurityGroup xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
    <groupName>RangedPortsBySource</groupName>
</DeleteSecurityGroup>
```

Sample Response

```
<DeleteSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</DeleteSecurityGroupResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress

DeregisterImage

The `DeregisterImage` operation deregisters an AMI. Once deregistered, instances of the AMI may no longer be launched.

Request Parameters

The following table describes the request parameters for `DeregisterImage`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
imageId	Unique ID of a machine image, returned by a call to RegisterImage or DescribeImages.	Yes	`xsd:string`

Response Tags

The following table describes the default response tags included in `DeregisterImage` responses.

Element Name	Definition
return	`true` if deregistration succeeded, otherwise `false`.

Sample Request

```
<DeregisterImage xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
    <imageId>ami-61a54008</imageId>
```

```
</DeregisterImage>
```

Sample Response

```
<DeregisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</DeregisterImageResponse>
```

Related Operations

- RegisterImage
- DescribeImages

DescribeImageAttribute

The `DescribeImageAttribute` operation returns information about an attribute of an AMI.

Request Parameters

The following table describes the request parameters for `DescribeImageAttribute`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
imageId	ID of the AMI for which an attribute will be described.	Yes	`xsd:string`
launchPermission	Describes launch permissions of the AMI.	Yes	`ec2:EmptyElementType`

Response Tags

The following table describes the default response tags included in `DescribeImageAttribute` responses.

Element Name	Definition
imageId	ID of the AMI of which parameters are being described.
launchPermission	Launch permissions of the AMI.

Sample Request

```
<DescribeImageAttribute xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageId>ami-61a54008</imageId>
  <launchPermission />
</DescribeImageAttribute>
```

Sample Response

```
<DescribeImageAttributeResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageId>ami-61a54008</imageId>
  <launchPermission>
    <item>
      <group>all</group>
    </item>
    <item>
      <userId>495219933132</userId>
    </item>
  </launchPermission>
</DescribeImageAttributeResponse>
```

Related Operations

- DescribeImages
- ModifyImageAttribute
- ResetImageAttribute

DescribeImages

The `DescribeImages` operation returns information about AMIs available for use by the user. This includes both public AMIs (those available for any user to launch) and private AMIs (those owned by the user making the request and those owned by other users that the user making the request has explicit launch permissions for).

The list of AMIs returned can be modified via optional lists of AMI IDs, owners or users with launch permissions. If all three optional lists are empty all AMIs the user has launch permissions for are returned. Launch permissions fall into three categories:

Launch Permission	Description
public	The `all` group has launch permissions for the AMI. All users have launch permissions for these AMIs.
explicit	The owner of the AMIs has granted a specific user launch permissions for the AMI.
implicit	A user has implicit launch permissions for all AMIs he or she owns.

If one or more of the lists are specified the result set is the intersection of AMIs matching the criteria of the indivdual lists.

Providing the list of AMI IDs requests information for those AMIs only. If no AMI IDs are provided, information of all relevant AMIs will be returned. If an AMI is specified that does not exist a fault is returned. If an AMI is specified that exists but the user making the request does not have launch permissions for, then that AMI will not be included in the returned results.

Providing the list of owners requests information for AMIs owned by the specified owners only. Only AMIs the user has launch permissions for are returned. The items of the list may be account ids for AMIs owned by users with those account ids, *`amazon`* for AMIs owned by Amazon or *`self`* for AMIs owned by the user making the request.

The executable list may be provided to request information for AMIs that only the specified users have launch permissions for. The items of the list may be account ids for AMIs owned by the user making the request that the users with the specified account ids have explicit launch permissions for, *self* for AMIs the user making the request has explicit launch permissions for or *all* for public AMIs.

Deregistered images will be included in the returned results for an unspecified interval subsequent to deregistration.

Request Parameters

The following table describes the request parameters for DescribeImages. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
imageSet	AMI IDs to describe.	Yes (but may be empty)	xsd:string[]
ownersSet	Owners of AMIs to describe	Yes (but may be empty)	xsd:string[]
executableBySet	Describe AMIs that the specified users have launch permissions for	Yes (but may be empty)	xsd:string[]

Response Tags

The following table describes the default response tags included in DescribeImages responses.

Element Name	Definition
imagesSet	A list of image descriptions

Sample Request

```
<DescribeImages xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <executableBySet>
    <item>
      <user>all</user>
    </item>
  </executableBySet>
  <ownersSet />
  <imagesSet>
    <item>
      <imageId>ami-61a54008</imageId>
      <imageId>ami-72f53012</imageId>
    </item>
  </imagesSet>
</DescribeImages>
```

Sample Response

```
<DescribeImagesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imagesSet>
    <item>
      <imageId>ami-61a54008</imageId>
      <imageLocation>aes-ttylinux/image.manifest.xml</imageLocation>
      <imageState>available</imageState>
      <imageOwnerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</imageOwnerId>
      <isPublic>true</isPublic>
    </item>
  </imagesSet>
</DescribeImagesResponse>
```

Related Operations

- DescribeInstances
- DescribeImageAttribute

DescribeInstances

The `DescribeInstances` operation returns information about instances owned by the user making the request.

An optional list of instance IDs may be provided to request information for those instances only. If no instance IDs are provided, information of all relevant instances information will be returned. If an instance is specified that does not exist a fault is returned. If an instance is specified that exists but is not owned by the user making the request, then that instance will not be included in the returned results.

Recently terminated instances will be included in the returned results for a small interval subsequent to their termination. This interval is typically of the order of one hour.

Request Parameters

The following table describes the request parameters for `DescribeInstances`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
instancesSet	Set of instances IDs to get the status of.	Yes (but may be empty)	`xsd:string[]`

Response Tags

The following table describes the default response tags included in `DescribeInstances` responses.

Element Name	Definition
reservationSet	A list of structures describing the status of all requested instances.

Sample Request

```
<DescribeInstances xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
    </item>
  </instancesSet>
</DescribeInstances>
```

Sample Response

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <reservationSet>
    <item>
      <reservationId>r-44a5402d</reservationId>
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupSet>
        <item>
          <groupId>default</groupId>
        </item>
      </groupSet>
      <instancesSet>
        <item>
          <instanceId>i-28a64341</instanceId>
          <imageId>ami-6ea54007</imageId>
          <instanceState>
            <code>0</code>
            <name>running</name>
          </instanceState>
          <dnsName>domU-12-31-33-00-00-5A.dc2.compute.amazonaws.com</dnsName>
          <keyName>example-key-name</keyName>
          <amiLaunchIndex>23</amiLaunchIndex>
        </item>
      </instancesSet>
    </item>
  </reservationSet>
</DescribeInstancesResponse>
```

Related Operations

- RunInstances
- TerminateInstances

DescribeKeyPairs

The `DescribeKeyPairs` operation returns information about keypairs available for use by the user making the request. Selected keypairs may be specified or the list may be left empty if information for all registered keypairs is required.

Request Parameters

The following table describes the request parameters for `DescribeKeyPairs`. Parameter names are case sensitive.

Element Name	Definition	Re-quired?	Type
keySet	Keypair IDs to describe.	Yes (but may be	`xsd:string[]`

Element Name	Definition	Re-quired?	Type
		empty)	

Response Tags

The following table describes the default response tags included in `DescribeKeyPairs` responses.

Element Name	Definition
keySet	A list of keypair descriptions

Sample Request

```
<DescribeKeyPairs xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <keySet>
    <item>
      <keyName>example-key-name</keyName>
    </item>
  </keySet>
</DescribeKeyPairs>
```

Sample Response

```
<DescribeKeyPairsResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <keySet>
    <item>
      <keyName>example-key-name</keyName>
<keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f</
keyFingerprint>
    </item>
  </keySet>
</DescribeKeyPairsResponse>
```

Related Operations

- CreateKeypair
- DeleteKeypair
- RunInstances

DescribeSecurityGroups

The `DescribeSecurityGroups` operation returns information about security groups owned by the user making the request.

An optional list of security group names may be provided to request information for those security groups only. If no security group names are provided, information of all security groups will be returned. If a group is specified that does not exist a fault is returned.

Request Parameters

The following table describes the request parameters for `DescribeSecurityGroups`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
securityGroupSet	List of security groups to describe.	Yes	`xsd:string[]`

Response Tags

The following table describes the default response tags included in `DescribeSecurityGroups` responses.

Element Name	Definition
securityGroupInfo	Information about security groups.

Sample Request

```
<DescribeSecurityGroups xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <securityGroupSet>
    <item>
      <groupName>WebServers</groupName>
    </item>
    <item>
      <groupName>RangedPortsBySource</groupName>
    </item>
  </securityGroupSet>
</DescribeSecurityGroups>
```

Sample Response

```
<DescribeSecurityGroupsResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <securityGroupInfo>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupName>WebServers</groupName>
      <groupDescription>Web</groupDescription>
      <ipPermissions>
        <item>
            <ipProtocol>tcp</ipProtocol>
          <fromPort>80</fromPort>
          <toPort>80</toPort>
          <groups/>
          <ipRanges>
            <item>
              <cidrIp>0.0.0.0/0</cidrIp>
            </item>
          </ipRanges>
        </item>
      </ipPermissions>
    </item>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupName>RangedPortsBySource</groupName>
      <groupDescription>A</groupDescription>
```

```
      <ipPermissions>
         <item>
          <ipProtocol>tcp</ipProtocol>
          <fromPort>6000</fromPort>
          <toPort>7000</toPort>
          <groups/>
          <ipRanges/>
        </item>
      </ipPermissions>
    </item>
  </securityGroupInfo>
</DescribeSecurityGroupsResponse>
```

Related Operations

- CreateSecurityGroup
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

GetConsoleOutput

The `GetConsoleOutput` operation retrieves console output that has been posted for the specified instance.

Instance console output is buffered and posted shortly after instance boot, reboot and once the instance is terminated. Only the most recent 64 KB of posted output is available. Console output is available for at least 1 hour after the most recent post.

Request Parameters

The following table describes the request parameters for `GetConsoleOutput`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
An instance ID returned from a previous call to `RunInstances`.	Yes	`xsd:string`	

Response Tags

The following table describes the default response tags included in `GetConsoleOutput` responses.

Element Name	Definition
`instanceId`	The instance ID.
`timestamp`	The time the output was last updated.
`output`	The console output, Base64 encoded.

Sample Request

```
<GetConsoleOutput xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
      <instanceId>i-28a64341</instanceId>
</GetConsoleOutput>
```

Sample Response

```
<GetConsoleOutputResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <instanceId>i-28a64341</instanceId>
  <timestamp>2007-01-03 15:00:00</timestamp>
<output>TGludXggdmVyc2lvbiAyLjYuMTYteGVuVSAoYnVpbGRlckBwYXRjaGJhdC5hbWF6b25zY
SkgKGdj
YyB2ZXJzaW9uIDQuMC4xIDIwMDUwNzI3IChSZWQgSGF0IDQuMC4xLTUpKSAjMSBTTVAgVGh1IE9j
dCAyNiAwODo0MToyNiBTQVNUIDIwMDYKQklPUy1wcm92aWRlZCBwaHlzaWNhbCBSQU0gbWFwOgpY
ZW46IDAwMDAwMDAwMDAwMDAwMDAgLSAwMDAwMDAwMDZhNDAwMDAwICh1c2FibGUpCjk4ME1CIEhJ
R0hNRU0gYXZhaWxhYmxlLgo3MjdNQiBMT1dNRU0gYXZhaWxhYmxlLgpOWCAoRXhlY3V0ZSBEaXNh
YmxlKSBwcm90ZWN0aW9uOiBhY3RpdmUKSVJRIGxvY2t1cCBkZXRlY3Rpb24gZGlzYWJsZWQKQnVp
bHQgMSB6b25lbGlzdHMKS2VybmVsIGNvbW1hbmQgbGluZTogcm9vdD0vZGV2L3NkYTEgcm8gNApF
bmFibGluZyBmYXN0IEZQVSBzYXZlIGFuZCByZXN0b3JlLi4uIGRvbmUuCg==</output>
</GetConsoleOutputResponse>
```

ModifyImageAttribute

The `ModifyImageAttribute` operation modifies an attribute of an AMI.

Currently the only attribute supported is launchPermission. By modifying this attribute it is possible to make an AMI public or to grant specific users launch permissions for the AMI. To make the AMI public add the `group=all` attribute item. To grant launch permissions for a specific user add a `userId=<userid>` attribute item.

Request Parameters

The following table describes the request parameters for `ModifyImageAttribute`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
imageId	AMI ID to modify an attribute on.	Yes	`xsd:string`
launchPermission	Adds or removes launch permissions for the AMI.	Yes	`ec2:LaunchPermissionOperationType`

Response Tags

The following table describes the default response tags included in `ModifyImageAttribute` responses.

Element Name	Definition
return	`true` if the operation succeeded, otherwise `false`.

Sample Request

```
<ModifyImageAttribute xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageId>ami-61a54008</imageId>
  <launchPermission>
    <add>
      <item>
        <group>all</group>
      </item>
      <item>
        <userId>495219933132</userId>
      </item>
    </add>
  <launchPermission>
</ModifyImageAttribute>
```

Sample Response

```
<ModifyImageAttributeResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</ModifyImageAttributeResponse>
```

Related Operations

- ResetImageAttribute
- DescribeImageAttribute

RebootInstances

The `RebootInstances` operation requests a reboot of one or more instances. This operation is asynchronous; it only queues a request to reboot the specified instance(s). The operation will succeed provided the instances are valid and belong to the user. Terminated instances will be ignored.

Request Parameters

The following table describes the request parameters for `RebootInstances`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
instancesSet	One or more instance IDs returned from previous calls to `RunInstances`.	Yes	`xsd:string[]`

Response Tags

The following table describes the default response tags included in `RebootInstances` responses.

Element Name	Definition
result	An indication of whether the request was successful.

Sample Request

```
<RebootInstances xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
    <instancesSet>
      <item>
        <instanceId>i-28a64341</instanceId>
      </item>
    </instancesSet>
</RebootInstances>
```

Sample Response

```
<RebootInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
      <boolean>true<boolean>
</RebootInstancesResponse>
```

RegisterImage

The `RegisterImage` operation registers an AMI with Amazon EC2. Images must be registered before they can be launched.

Each AMI is associated with an unique ID which is provided by the EC2 service via the Registerimage operation. As part of the registration process, Amazon EC2 will retrieve the specified image manifest from Amazon S3 and verify that the image is owned by the user requesting image registration.

The image manifest is retrieved once and stored within the Amazon EC2 network. Any modifications to an image in Amazon S3 invalidate this registration. If you do have to make changes and upload a new image deregister the previous image and register the new image.

Request Parameters

The following table describes the request parameters for `RegisterImage`. Parameter names are case sensitive.

Element Name	Definition	Re-quired?	Type
imageLocation	Full path to your AMI manifest in Amazon S3 storage.	Yes	`xsd:string`

Response Tags

The following table describes the default response tags included in `RegisterImage` responses.

Element Name	Definition	Type
imageId	Unique ID of the newly registered machine image.	`xsd:string`

Sample Request

```
<RegisterImage xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageLocation>/mybucket/myimage.manifest.xml</imageLocation>
</RegisterImage>
```

Sample Response

```
<RegisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageId>ami-61a54008</imageId>
</RegisterImageResponse>
```

Related Operations

- DescribeImages
- DeregisterImage

ResetImageAttribute

The `ResetImageAttribute` operation resets an attribute of an AMI to its default value.

Request Parameters

The following table describes the request parameters for `ResetImageAttribute`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
imageId	ID of the AMI on which the attribute will be reset.	Yes	`xsd:string`
launchPermission	Resets the AMI's launch permissions. All public and explicit launch permissions for the AMI are revoked.	Yes	`ec2:EmptyElementType`

Response Tags

The following table describes the default response tags included in `ResetImageAttribute` responses.

Element Name	Definition
return	`true` if the operation succeeded, otherwise `false`.

Sample Request

```
<ResetImageAttribute xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageId>ami-61a54008</imageId>
  <launchPermission />
</ResetImageAttribute>
```

Sample Response

```
<ResetImageAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</ResetImageAttributeResponse>
```

Related Operations

- ModifyImageAttribute
- DescribeImageAttribute

RevokeSecurityGroupIngress

The `RevokeSecurityGroupIngress` operation revokes existing permissions that were previously granted to a security group. The permissions to revoke must be specified using the same values originally used to grant the permission.

Permissions are specified in terms of the IP protocol (TCP, UDP or ICMP), the source of the request (by IP range or an Amazon EC2 user-group pair), source and destination port ranges (for TCP and UDP), and ICMP codes and types (for ICMP).

> **Note**
> Changes are anticipated in this API that may restrict further what is allowable. Please consult the section called "Anticipated API changes" for more details.

Permission changes are propagated to instances within the security group being modified as quickly as possible. However, a small delay is likely, depending on the number of instances that are members of the indicated group.

Request Parameters

The following table describes the request parameters for `RevokeSecurityGroupIngress`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
userId	AWS Access Key ID.	Yes	`xsd:string`
groupName	Name of the group to modify.	Yes	`xsd:string`
ipPermissions	Set of permissions to remove from the group.	Yes	`ec2:IpPermissionType[]`

Response Tags

The following table describes the default response tags included in `RevokeSecurityGroupIngress` responses.

Element Name	Definition
return	`true` if permissions successfully revoked.

Sample Request

```
<RevokeSecurityGroupIngress xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
    <userId/>
    <groupName>RangedPortsBySource</groupName>
    <ipPermissions>
        <item>
            <ipProtocol>tcp</ipProtocol>
            <fromPort>6000</fromPort>
            <toPort>7000</toPort>
            <groups/>
            <ipRanges/>
        </item>
    </ipPermissions>
</RevokeSecurityGroupIngress>
```

Sample Response

```
<RevokeSecurityGroupIngressResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</RevokeSecurityGroupIngressResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- DeleteSecurityGroup

RunInstances

The `RunInstances` operation launches a specified number of instances.

A call to `RunInstances` is guaranteed to start no fewer than the requested minimum for each AMI specified. If there is insufficient capacity available then no instances will be started. Amazon EC2 will make a best effort attempt to satisfy the requested maximum values. If there is capacity to cover the specified minimum values but not the maximum values then instances of each image specified will be launched in a round robin fashion.

As an example, consider a request to launch two images (A and B), with minimum and maximum values of (5,10) and (20, 40) respectively.

If there is sufficient capacity for less than 25 instances then no instances will be launched (since the minimums of 5 and 20 cannot both be satisfied).

If there is capacity available for only 30 instances then 5 instances of A and 20 instances of B will be launched. The remaining 5 instances will be allocated in round robin fashion.

Every instance is launched in a security group. This may be specified as part of the launch request. If a security group is not indicated then instances are started in a the default security group.

An optional keypair ID may be provided for each image in the launch request. All instances that are created from images for which this is provided will have access to the associated public key at boot time (detailed below). This key may be used to provide secure access to an instance of an image on a

per-instance basis. Amazon EC2 public images make use of this functionality to provide secure passwordless access to instances (and launching those images without a keypair ID will leave them inaccessible).

The public key material is made available to the instance at boot time by placing it in a file named `openssh_id.pub` on a logical device that is exposed to the instance as `/dev/sda2` (the ephemeral store). The format of this file is suitable for use as an entry within `~/.ssh/authorized_keys` (the OpenSSH format). This can be done at boot time (as part of `rclocal`, for example) allowing for secure password-less access. As the need arises, other formats will also be considered.

Optional user data may be provided in the launch request. All instances comprising the launch request have access to this data (see the section called “Using Instance Data” for details).

Request Parameters

The following table describes the request parameters for `RunInstances`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
instancesSet	Description of the instances to launch.	Yes	ec2:RunInstanceItemType[]
groupSet	Description of the security groups to associate the instances with.	Yes	ec2:GroupSetType[]
userData	The user data available to the launched instances.	No	ec2:UserDataType

Response Tags

The following table describes the default response tags included in `RunInstances` responses.

Element Name	Definition
RunInstancesResponse	Status information about the instances started.

Sample Request

```
<RunInstances xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
    <instancesSet>
        <item>
            <imageId>ami-60a54009</imageId>
            <minCount>1</minCount>
            <maxCount>3</maxCount>
            <keyName>example-key-name</keyName>
        </item>
    </instancesSet>
    <groupSet/>
    <userData version="1.0" encod-
ing="base64"><data>"VGhpcyBpcyBiYXNlIDY0IQ==</data></userData>
</RunInstances>
```

Sample Response

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>495219933132</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
      <amiLaunchIndex>0</amiLaunchIndex>
    </item>
    <item>
      <instanceId>i-2bc64242</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
      <amiLaunchIndex>1</amiLaunchIndex>
    </item>
    <item>
      <instanceId>i-2be64332</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
      <amiLaunchIndex>2</amiLaunchIndex>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

Related Operations

- DescribeInstances
- TerminateInstances
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DescribeSecurityGroups

TerminateInstances

The `TerminateInstances` operation shuts down one or more instances. This operation is idempotent and terminating an instance that is in the process of shutting down (or already terminated) will succeed.

Terminated instances remain visible for a short period of time (approximately one hour) after termination, after which their instance ID is invalidated.

Request Parameters

The following table describes the request parameters for `TerminateInstances`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
instancesSet	One or more instance IDs returned from previous calls to `RunInstances`.	Yes	`xsd:string[]`

Response Tags

The following table describes the default response tags included in `TerminateInstances` responses.

Element Name	Definition
instancesSet	A complex type containing describing the current and new state of each instance specified.

Sample Request

```
<TerminateInstances xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
    </item>
  </instancesSet>
</TerminateInstances>
```

Sample Response

```
<TerminateInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
      <shutdownState>
        <code>32</code>
        <name>shutting-down</name>
      </shutdownState>
      <previousState>
        <code>16</code>
        <name>running</name>
      </previousState>
    </item>
  </instancesSet>
</TerminateInstancesResponse>
```

Related Operations

- DescribeInstances

EC2 Query API

The Amazon EC2 API consists of web service operations for every task the service can perform. This section describes each operation in detail.

Common Query Parameters

Request Parameters

All Query operations share a set of common parameters that must be present in each call:

Parameter Name	Description	Example Value
Action	Indicates the action to perform.	RunInstances
Version	The API version to use, as specified in the WSDL.	2007-01-03
AWSAccessKeyId	The Access Key ID for the request sender. This identifies the account which will be charged for usage of the service. The account with which the Access Key ID is associated must be signed up for EC2, or requests will not be accepted.	10QMXFEV71ZS32XQFTR2
Timestamp	The date and time at which the request is signed, in the format YYYY-MM-DDThh:mm:ssZ, as specified in the ISO 8601 standard.	2006-07-07T15:04:56Z
Expires	The date and time at which the signature included in the request expires, in the format YYYY-MM-DDThh:mm:ssZ, as specified in the ISO 8601 standard.	2006-07-07T15:04:56Z
Signature	A request signature is calculated as explained in Request Authentication.	Qnpl4Qk/7tINHzfXCiT7VbBatDA=
SignatureVersion	A value of 0 or 1 indicates the method chosen to construct the string to be signed. Currently, only a value of 1 is valid.	1

Note
The *Timestamp* parameter can be used instead of *Expires*. Requests must include either *Timestamp* or *Expires*, but cannot contain both.

Parameter values must be URL-encoded. This is true for any Query parameter passed to EC2 and is typically necessary in the *Signature* parameter. Some clients do this automatically, but this is not the norm.

By Function

Operations

Images

- RegisterImage
- DescribeImages
- DeregisterImage

Instances

- RunInstances
- DescribeInstances
- TerminateInstances

Keypairs

- CreateKeyPair
- DescribeKeyPairs
- DeleteKeyPair

Image Attributes

- ModifyImageAttribute
- DescribeImageAttribute
- ResetImageAttribute

Security Groups

- CreateSecurityGroup
- DescribeSecurityGroups
- DeleteSecurityGroup
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress

AuthorizeSecurityGroupIngress

The `AuthorizeSecurityGroupIngress` operation adds permissions to a security group.

Permissions are specified in terms of the IP protocol (TCP, UDP or ICMP), the source of the request (by IP range or an Amazon EC2 user-group pair), source and destination port ranges (for TCP and UDP), and ICMP codes and types (for ICMP). When authorizing ICMP, `-1` may be used as a wildcard in the type and code fields.

Permission changes are propagated to instances within the security group being modified as quickly as possible. However, a small delay is likely, depending on the number of instances that are members of

the indicated group.

When authorizing a user/group pair permission, *GroupName*, *SourceSecurityGroupName* and *SourceSecurityGroupOwnerId* must be specified. When authorizing a CIDR IP permission, *GroupName*, *IpProtocol*, *FromPort*, *ToPort* and *CidrIp* must be specified. Mixing these two types of parameters is not allowed.

Request Parameters

The following table describes the request parameters for `AuthorizeSecurityGroupIngress`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
GroupName	Name of the group to modify.	Yes	`string`
SourceSecurityGroupName	Name of security group to authorize access to when operating on a user/group pair.	When authorizing user/group pair permission.	`string`
SourceSecurityGroupOwnerId	Owner of security group to authorize access to when operating on a user/group pair.	When authorizing user/group pair permisison.	`string`
IpProtocol	IP protocol to authorize access to when operating on a CIDR IP. Valid values are `tcp`, `udp` and `icmp`.	When authorizing CIDR IP permission.	`string`
FromPort	Bottom of port range to authorize access to when operating on a CIDR IP. This contains the ICMP type if ICMP is being authorized.	When authorizing CIDR IP permission.	`int`
ToPort	Top of port range to authorize access to when operating on a CIDR IP. This contains the ICMP code if ICMP is being authorized.	When authorizing CIDR IP permission.	`int`
CidrIp	CIDR IP range to authorize access to when operating on a CIDR IP.	When authorizing CIDR IP per-	`string`

Element Name	Definition	Re-quired?	Type
		mission.	

Response Tags

The following table describes the default response tags included in `AuthorizeSecurityGroupIngress` responses.

Element Name	Definition
return	`true` if permissions successfully added.

Sample Request

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&IpProtocol=tcp
&FromPort=80
&ToPort=80
&CidrIp=0.0.0.0/0
&...auth parameters...
```

Sample Response

```
<AuthorizeSecurityGroupIngressResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</AuthorizeSecurityGroupIngressResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

CreateKeyPair

The `CreateKeyPair` operation creates a new 2048 bit RSA keypair and returns a unique ID that can be used to reference this keypair when launching new instances.

Request Parameters

The following table describes the request parameters for `CreateKeyPair`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
KeyName	A unique name for this key.	Yes	string

Response Tags

The following table describes the default response tags included in CreateKeyPair responses.

Element Name	Definition
keyName	The key name provided in the original request.
KeyFingerprint	A SHA-1 digest of the DER encoded private key.
KeyMaterial	An unencrypted PEM encoded RSA private key.

Sample Request

```
https://ec2.amazonaws.com/
?Action=CreateKeyPair
&KeyName=example-key-name
&...auth parameters...
```

Sample Response

```
<CreateKeyPairResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <keyName>example-key-name</keyName>
<keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f</
keyFingerprint>
  <keyMaterial>-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBuLFg5ujHrtm1jnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFxTHgElQiJLChp
HungXQ29VTc8rc1bW0lkdi23OH5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKCOVUR+jx71Sg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjGlUKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3LpGZyq/
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYGtGSBMpTRP5hnbzzuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAY1tsiUsIwDl5
91CXirkYGuVfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVD1pM
ZNUJs7rw9gZRTrf7LylaJ58kOcyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1kGQI7
3wcfgt5ecIu4TZf0OE9IHjn+2eRlsrjBdeORi7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhHOux5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLGCwU9h9HlO9mKAc2m8Cm1
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkpHi2EISh1uRkhxljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAy1/EDNTY5OSIJU+0KFmQbyhsbm
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfqGoQc7+sAc8vmKkesnF7CqCEKDyF/dhrxYdQKB
gC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeEO+uZ58/BGKOIGHByHBDiXtzMhdJr15HTYjxK7OgTZm
gK+8zp4L9IbvLGDMJO8vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+Zrvm1F0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0gij1houRe
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----</keyMaterial>
</CreateKeyPairResponse>
```

Related Operations

- DescribeKeyPairs
- DeleteKeyPair
- RunInstances

CreateSecurityGroup

The `CreateSecurityGroup` operation creates a new security group.

Every instance is launched in a security group. If none is specified as part of the launch request then instances are launched in the default security group. Instances within the same security group have unrestricted network access to one another. Instances will reject network access attempts from other instances in a different security group. As the owner of instances you may grant or revoke specific permissions using the AuthorizeSecurityGroupIngress and RevokeSecurityGroupIngress operations.

Request Parameters

The following table describes the request parameters for `CreateSecurityGroup`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
GroupName	Name for the new security group.	Yes	`string`
GroupDescription	Description of the new security group.	Yes	`string`

Response Tags

The following table describes the default response tags included in `CreateSecurityGroup` responses.

Element Name	Definition
return	`true` if call succeeded.

Sample Request

```
https://ec2.amazonaws.com/
?Action==CreateSecurityGroup
&GroupName=WebServers
&GroupDescription=Web
&...auth parameters...
```

Sample Response

```
<CreateSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</CreateSecurityGroupResponse>
```

Related Operations

- RunInstances
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

DeleteKeyPair

The `DeleteKeyPair` operation deletes a keypair.

Request Parameters

The following table describes the request parameters for `DeleteKeyPair`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
KeyName	Name of the keypair to delete.	Yes	`string`

Response Tags

The following table describes the default response tags included in `DeleteKeyPair` responses.

Element Name	Definition
return	`true` if the key was successfully deleted.

Sample Request

```
https://ec2.amazonaws.com/
?Action=DeleteKeyPair
&KeyName=example-key-name
&...auth parameters...
```

Sample Response

```
<DeleteKeyPair xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</DeleteKeyPair>
```

Related Operations

- CreateKeyPair
- DescribeKeyPairs

DeleteSecurityGroup

The `DeleteSecurityGroup` operation deletes a security group.

If an attempt is made to delete a security group and any instances exist that are members of that group a fault is returned.

Request Parameters

The following table describes the request parameters for `DeleteSecurityGroup`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
GroupName	Name of the security group to delete.	Yes	`string`

Response Tags

The following table describes the default response tags included in `DeleteSecurityGroup` responses.

Element Name	Definition
return	`true` if group deleted.

Sample Request

```
https://ec2.amazonaws.com/
?Action=DeleteSecurityGroup
&GroupName=RangedPortsBySource
&...auth parameters...
```

Sample Response

```
<DeleteSecurityGroupResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</DeleteSecurityGroupResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress

DeregisterImage

The `DeregisterImage` operation deregisters an AMI. Once deregistered, instances of the AMI may no longer be launched.

Request Parameters

The following table describes the request parameters for `DeregisterImage`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
ImageId	Unique ID of a machine image, returned by a call to RegisterImage or DescribeImages.	Yes	`string`

Response Tags

The following table describes the default response tags included in `DeregisterImage` responses.

Element Name	Definition
return	`true` if deregistration succeeded, otherwise `false`.

Sample Request

```
https://ec2.amazonaws.com/
?Action=DeregisterImage
&ImageId=ami-61a54008
&...auth parameters...
```

Sample Response

```
<DeregisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</DeregisterImageResponse>
```

Related Operations

- RegisterImage
- DescribeImages

DescribeImageAttribute

The `DescribeImageAttribute` operation returns information about an attribute of an AMI.

Request Parameters

The following table describes the request parameters for `DescribeImageAttribute`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
ImageId	Id of the AMI for which an attribute will be described.	Yes	`string`

Element Name	Definition	Required?	Type
Attribute	Specifies the attribute to describe. Currently, only `launchPermission` is supported.	Yes	`string`

Response Tags

The following table describes the default response tags included in `DescribeImageAttribute` responses.

Element Name	Definition
imageId	ID of the AMI of which parameters are being described.
launchPermission	Launch permissions of the AMI.

Sample Request

```
https://ec2.amazonaws.com/
?Action=DescribeImageAttribute
&ImageId=ami-61a54008
&Attribute=launchPermission
&...auth parameters...
```

Sample Response

```
<DescribeImageAttributeResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageId>ami-61a54008</imageId>
  <launchPermission>
    <item>
      <group>all</group>
    </item>
    <item>
      <userId>495219933132</userId>
    </item>
  </launchPermission>
</DescribeImageAttributeResponse>
```

Related Operations

- DescribeImages
- ModifyImageAttribute
- ResetImageAttribute

DescribeImages

The `DescribeImages` operation returns information about AMIs available for use by the user. This includes both public AMIs (those available for any user to launch) and private AMIs (those owned by the user making the request and those owned by other users that the user making the request has explicit

launch permissions for).

The list of AMIs returned can be modified via optional lists of AMI IDs, owners or users with launch permissions. If all three optional lists are empty all AMIs the user has launch permissions for are returned. Launch permissions fall into three categories:

Launch Permission	Description
public	The `all` group has launch permissions for the AMI. All users have launch permissions for these AMIs.
explicit	The owner of the AMIs has granted a specific user launch permissions for the AMI.
implicit	A user has implicit launch permissions for all AMIs he or she owns.

If one or more of the lists are specified the result set is the intersection of AMIs matching the criteria of the individual lists.

Providing the list of AMI IDs requests information for those AMIs only. If no AMI IDs are provided, information of all relevant AMIs will be returned. If an AMI is specified that does not exist a fault is returned. If an AMI is specified that exists but the user making the request does not have launch permissions for, then that AMI will not be included in the returned results.

Providing the list of owners requests information for AMIs owned by the specified owners only. Only AMIs the user has launch permissions for are returned. The items of the list may be account ids for AMIs owned by users with those account ids, *`amazon`* for AMIs owned by Amazon or *`self`* for AMIs owned by the user making the request.

The executable list may be provided to request information for AMIs that only the specified users have launch permissions for. The items of the list may be account ids for AMIs owned by the user making the request that the users with the specified account ids have explicit launch permissions for, *`self`* for AMIs the user making the request has explicit launch permissions for or *`all`* for public AMIs.

Deregistered images will be included in the returned results for an unspecified interval subsequent to deregistration.

Request Parameters

The following table describes the request parameters for `DescribeImages`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
`ImageId.n`	A list of image descriptions	No	`string`
`Owner.n`	Owners of AMIs to describe	No	`string`
`ExecutableBy.n`	Describe AMIs that the specified users have launch permissions for	No	`string`

Response Tags

The following table describes the default response tags included in `DescribeImages` responses.

Element Name	Definition
imagesSet	A list of image descriptions

Sample Request

```
https://ec2.amazonaws.com/
?Action=DescribeImages
&ImageId.1=ami-61a54008
&...auth parameters...
```

Sample Response

```
<DescribeImagesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imagesSet>
    <item>
      <imageId>ami-61a54008</imageId>
      <imageLocation>aes-ttylinux/image.manifest.xml</imageLocation>
      <imageState>available</imageState>
      <imageOwnerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</imageOwnerId>
      <isPublic>false</isPublic>
    </item>
  </imagesSet>
</DescribeImagesResponse>
```

Related Operations

- DescribeInstances
- DescribeImageAttribute

DescribeInstances

The `DescribeInstances` operation returns information about instances owned by the user making the request.

An optional list of instance IDs may be provided to request information for those instances only. If no instance IDs are provided, information of all relevant instances information will be returned. If an instance is specified that does not exist a fault is returned. If an instance is specified that exists but is not owned by the user making the request, then that instance will not be included in the returned results.

Recently terminated instances will be included in the returned results for a small interval subsequent to their termination. This interval is typically of the order of one hour.

Request Parameters

The following table describes the request parameters for `DescribeInstances`. Parameter names are case sensitive.

Element Name	Definition	Re-quired?	Type
InstanceId.n	Set of instances IDs to get the status of.	No	`string`

Response Tags

The following table describes the default response tags included in `DescribeInstances` responses.

Element Name	Definition
reservationSet	A list of structures describing the status of all requested instances.

Sample Request

```
https://ec2.amazonaws.com/
?Action=DescribeInstances
&InstanceId.1=i-28a64341
&...auth parameters...
```

Sample Response

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <reservationSet>
    <item>
      <reservationId>r-44a5402d</reservationId>
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupSet>
        <item>
          <groupId>default</groupId>
        </item>
      </groupSet>
      <instancesSet>
        <item>
          <instanceId>i-28a64341</instanceId>
          <imageId>ami-6ea54007</imageId>
          <instanceState>
            <code>0</code>
            <name>running</name>
          </instanceState>
          <dnsName>domU-12-31-33-00-00-5A.dc2.compute.amazonaws.com</dnsName>
          <keyName>example-key-name</keyName>
        </item>
      </instancesSet>
    </item>
  </reservationSet>
</DescribeInstancesResponse>
```

Related Operations

- RunInstances
- TerminateInstances

DescribeKeyPairs

The `DescribeKeyPairs` operation returns information about keypairs available for use by the user making the request. Selected keypairs may be specified or the list may be left empty if information for all registered keypairs is required.

Request Parameters

The following table describes the request parameters for `DescribeKeyPairs`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
KeyName.n	Keypair IDs to describe.	No	`string`

Response Tags

The following table describes the default response tags included in `DescribeKeyPairs` responses.

Element Name	Definition
keySet	A list of keypair descriptions

Sample Request

```
https://ec2.amazonaws.com/
?Action=DescribeKeyPairs
&KeyName.1=example-key-name
&...auth parameters...
```

Sample Response

```
<DescribeKeyPairsResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <keySet>
    <item>
      <keyName>example-key-name</keyName>
<keyFingerprint>1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f</
keyFingerprint>
    </item>
  </keySet>
</DescribeKeyPairsResponse>
```

Related Operations

- CreateKeypair
- DeleteKeypair
- RunInstances

DescribeSecurityGroups

The `DescribeSecurityGroups` operation returns information about security groups owned by the user making the request.

An optional list of security group names may be provided to request information for those security groups only. If no security group names are provided, information of all security groups will be returned. If a group is specified that does not exist a fault is returned.

Request Parameters

The following table describes the request parameters for `DescribeSecurityGroups`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
GroupName.n	List of security groups to describe.	No	`string`

Response Tags

The following table describes the default response tags included in `DescribeSecurityGroups` responses.

Element Name	Definition
securityGroupInfo	Information about security groups.

Sample Request

```
https://ec2.amazonaws.com/
?Action=DescribeSecurityGroups
&GroupName.1=WebServers
&GroupName.2=RangedPortsBySource
&...auth parameters...
```

Sample Response

```
<DescribeSecurityGroupsResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <securityGroupInfo>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupName>WebServers</groupName>
      <groupDescription>Web</groupDescription>
      <ipPermissions>
        <item>
            <ipProtocol>tcp</ipProtocol>
          <fromPort>80</fromPort>
          <toPort>80</toPort>
          <groups/>
          <ipRanges>
            <item>
              <cidrIp>0.0.0.0/0</cidrIp>
            </item>
          </ipRanges>
         </item>
      </ipPermissions>
    </item>
    <item>
      <ownerId>UYY3TLBUXIEON5NQVUUX6OMPWBZIQNFM</ownerId>
      <groupName>RangedPortsBySource</groupName>
      <groupDescription>A</groupDescription>
      <ipPermissions>
         <item>
          <ipProtocol>tcp</ipProtocol>
          <fromPort>6000</fromPort>
          <toPort>7000</toPort>
          <groups/>
          <ipRanges/>
        </item>
```

```
      </ipPermissions>
    </item>
  </securityGroupInfo>
</DescribeSecurityGroupsResponse>
```

Related Operations

- CreateSecurityGroup
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DeleteSecurityGroup

GetConsoleOutput

The `GetConsoleOutput` operation retrieves console output that has been posted for the specified instance.

Instance console output is buffered and posted shortly after instance boot, reboot and once the instance is terminated. Only the most recent 64 KB of posted output is available. Console output is available for at least 1 hour after the most recent post.

Request Parameters

The following table describes the request parameters for `GetConsoleOutput`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
`InstanceId`	An instance ID returned from a previous call to `RunInstances`.	Yes	`string`

Response Tags

The following table describes the default response tags included in `GetConsoleOutput` responses.

Element Name	Definition
`instanceId`	The instance ID.
`timestamp`	The time the output was last updated.
`output`	The console output, Base64 encoded.

Sample Request

```
https://ec2.amazonaws.com/
?Action=GetConsoleOutput
&InstanceId=i-2ea64347
&...auth parameters...
```

Sample Response

```
<GetConsoleOutputResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <instanceId>i-28a64341</instanceId>
  <timestamp>2007-01-03 15:00:00</timestamp>
<output>TGludXggdmVyc2lvbiAyLjYuMTYteGVuVSAoYnVpbGRlckBwYXRjaGJhdC5hbWF6b25zY
SkgKGdj
YyB2ZXJzaW9uIDQuMC4xIDIwMDUwNzI3IChSZWQgSGF0IDQuMC4xLTUpKSAjMSBTTVAgVGh1IE9j
dCAyNiAwODo0MToyNiBTQVNUIDIwMDYKQklPUy1wcm92aWRlZCBwaHlzaWNhbCBSQU0gbWFwOgpY
ZW46IDAwMDAwMDAwMDAwMDAwMDAgLSAwMDAwMDAwMDZhNDAwMDAwICh1c2FibGUpCjk4ME1CIEhJ
R0hNRU0gYXZhaWxhYmxlLgo3MjdNQiBMT1dNRU0gYXZhaWxhYmxlLgpOWCAoRXhlY3V0ZSBEaXNh
YmxlKSBwcm90ZWN0aW9uOiBhY3RpdmUKSVJRIGxvY2t1cCBkZXRlY3Rpb24gZGlzYWJsZWQKQnVp
bHQgMSB6b25lbGlzdHMKS2VybmVsIGNvbW1hbmQgbGluZTogcm9vdD0vZGV2L3NkYTEgcm8gNApF
bmFibGluZyBmYXN0IEZQVSBzYXZlIGFuZCByZXN0b3JlLi4uIGRvbmUuCg==</output>
</GetConsoleOutputResponse>
```

ModifyImageAttribute

The `ModifyImageAttribute` operation modifies an attribute of an AMI.

Currently the only attribute supported is launchPermission. By modifying this attribute it is possible to make an AMI public or to grant specific users launch permissions for the AMI. To make the AMI public add the `group=all` attribute item. To grant launch permissions for a specific user add a `userId=<userid>` attribute item.

Request Parameters

The following table describes the request parameters for `ModifyImageAttribute`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
ImageId	AMI Id to modify an attribute on.	Yes	`string`
Attribute	Specifies the attribute to modify. Currently, only `launchPermission` is supported.	Yes	`string`
OperationType	Specifies the operation to perform on the attribute. Currently only `add` and `remove` are supported.	Yes	`string`
UserId.n	User ids to add to or remove from the `launchPermission` attribute.	With `launchPermission` attribute	`string`
UserGroup.n	User groups to add to or remove from the `launchPermission` attribute. Currently, only the `all` group is available, specifiying all Amazon EC2 users.	With `launchPermission` attribute	`string`

Response Tags

The following table describes the default response tags included in `ModifyImageAttribute` responses.

Element Name	Definition
return	`true` if the operation succeeded, otherwise `false`.

Sample Request

```
https://ec2.amazonaws.com/
?Action=ModifyImageAttribute
&ImageId=ami-61a54008
&Attribute=launchPermission
&OperationType=add
&Group.1=all
&UserId.1=495219933132
&...auth parameters...
```

Sample Response

```
<ModifyImageAttributeResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</ModifyImageAttributeResponse>
```

Related Operations

- ResetImageAttribute
- DescribeImageAttribute

RebootInstances

The `RebootInstances` operation requests a reboot of one or more instances. This operation is asynchronous; it only queues a request to reboot the specified instance(s). The operation will succeed provided the instances are valid and belong to the user. Terminated instances will be ignored.

Request Parameters

The following table describes the request parameters for `RebootInstance`. Parameter names are case sensitive.

Element Name	Definition	Re-quired?	Type
InstanceId.n	One or more instance IDs returned from previous calls to `RunInstances`.	Yes	`string`

Response Tags

The following table describes the default response tags included in `RebootInstances` responses.

Element Name	Definition
result	An indication of whether the request was successful.

Sample Request

```
https://ec2.amazonaws.com/
?Action=RebootInstances
&InstanceId.1=i-2ea64347
&InstanceId.2=i-21a64348
&...auth parameters...
```

Sample Response

```
<RebootInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
      <boolean>true<boolean>
</RebootInstancesResponse>
```

RegisterImage

The `RegisterImage` operation registers an AMI with Amazon EC2. Images must be registered before they can be launched.

Each AMI is associated with an unique ID which is provided by the EC2 service via the Registerimage operation. As part of the registration process, Amazon EC2 will retrieve the specified image manifest from Amazon S3 and verify that the image is owned by the user requesting image registration.

The image manifest is retrieved once and stored within the Amazon EC2 network. Any modifications to an image in Amazon S3 invalidate this registration. If you do have to make changes and upload a new image deregister the previous image and register the new image.

Request Parameters

The following table describes the request parameters for `RegisterImage`. Parameter names are case sensitive.

Element Name	Definition	Re-quired?	Type
`ImageLocation`	Full path to your AMI manifest in Amazon S3 storage.	Yes	`string`

Response Tags

The following table describes the default response tags included in `RegisterImage` responses.

Element Name	Definition	Type
`imageId`	Unique ID of the newly registered machine image.	`xsd:string`

Sample Request

```
https://ec2.amazonaws.com/
?Action=RegisterImage
&ImageLocation=mybucket-myimage.manifest.xml
&...auth parameters...
```

Sample Response

```
<RegisterImageResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <imageId>ami-61a54008</imageId>
</RegisterImageResponse>
```

Related Operations

- DescribeImages
- DeregisterImage

ResetImageAttribute

The `ResetImageAttribute` operation resets an attribute of an AMI to its default value.

Request Parameters

The following table describes the request parameters for `ResetImageAttribute`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
ImageId	Id of the AMI for which an attribute will be described.	Yes	`string`
Attribute	Specifies the attribute to reset. Currently, only `launchPermission` is supported. In the case of `launchPermission`, all public and explicit launch permissions for the AMI are revoked.	Yes	`string`

Response Tags

The following table describes the default response tags included in `ResetImageAttribute` responses.

Element Name	Definition
return	`true` if the operation succeeded, otherwise `false`.

Sample Request

```
https://ec2.amazonaws.com/
?Action=ResetImageAttribute
```

```
&ImageId=ami-61a54008
&Attribute=launchPermission
&...auth parameters...
```

Sample Response

```
<ResetImageAttributeResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</ResetImageAttributeResponse>
```

Related Operations

- ModifyImageAttribute
- DescribeImageAttribute

RevokeSecurityGroupIngress

The `RevokeSecurityGroupIngress` operation revokes existing permissions that were previously granted to a security group. The permissions to revoke must be specified using the same values originally used to grant the permission.

Permissions are specified in terms of the IP protocol (TCP, UDP or ICMP), the source of the request (by IP range or an Amazon EC2 user-group pair), source and destination port ranges (for TCP and UDP), and ICMP codes and types (for ICMP). When authorizing ICMP, `-1` may be used as a wildcard in the type and code fields.

Permission changes are propagated to instances within the security group being modified as quickly as possible. However, a small delay is likely, depending on the number of instances that are members of the indicated group.

When revoking a user/group pair permission, *`GroupName`*, *`SourceSecurityGroupName`* and *`SourceSecurityGroupOwnerId`* must be specified. When authorizing a CIDR IP permission, *`GroupName`*, *`IpProtocol`*, *`FromPort`*, *`ToPort`* and *`CidrIp`* must be specified. Mixing these two types of parameters is not allowed.

Request Parameters

The following table describes the request parameters for `RevokeSecurityGroupIngress`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
`GroupName`	Name of the group to modify.	Yes	`string`
`SourceSecurityGroupName`	Name of security group to revoke access to when operating on a user/group pair.	When revoking user/group pair permission.	`string`
`SourceSecurityGroupOwn-`	Owner of security group to revoke access to when operating on a user/group	When revok-	`string`

Element Name	Definition	Required?	Type
erId	pair.	ing user/ group pair permisison.	
IpProtocol	IP protocol to revoke access to when operating on a CIDR IP. Valid values are `tcp`, `udp` and `icmp`.	When revoking CIDR IP permission.	`string`
FromPort	Bottom of port range to revoke access to when operating on a CIDR IP. This contains the ICMP type if ICMP is being authorized.	When revoking CIDR IP permission.	`int`
ToPort	Top of port range to revoke access to when operating on a CIDR IP. This contains the ICMP code if ICMP is being authorized.	When revoking CIDR IP permission.	`int`
CidrIp	CIDR IP range to revoke access to when operating on a CIDR IP.	When revoking CIDR IP permission.	`string`

Response Tags

The following table describes the default response tags included in `RevokeSecurityGroupIngress` responses.

Element Name	Definition
return	`true` if permissions successfully revoked.

Sample Request

```
https://ec2.amazonaws.com/
?Action=AuthorizeSecurityGroupIngress
&IpProtocol=tcp
&FromPort=80
&ToPort=80
&CidrIp=0.0.0.0/0
&...auth parameters...
```

Sample Response

```
<RevokeSecurityGroupIngressResponse xm-
lns="http://ec2.amazonaws.com/doc/2007-01-03">
  <return>true</return>
</RevokeSecurityGroupIngressResponse>
```

Related Operations

- CreateSecurityGroup
- DescribeSecurityGroups
- AuthorizeSecurityGroupIngress
- DeleteSecurityGroup

RunInstances

The `RunInstances` operation launches a specified number of instances.

> **Note**
> The Query version of `RunInstances` only allows instances of a single AMI to be launched in one call. This is different from the SOAP API call of the same name but similar to the **ec2-run-instances** command line tool.

A call to `RunInstances` is guaranteed to start no fewer than the requested minimum. If there is insufficient capacity available then no instances will be started. Amazon EC2 will make a best effort attempt to satisfy the requested maximum values.

Every instance is launched in a security group. This may be specified as part of the launch request. If a security group is not indicated then instances are started in a the default security group.

An optional keypair ID may be provided for each image in the launch request. All instances that are created from images for which this is provided will have access to the associated public key at boot time (detailed below). This key may be used to provide secure access to an instance of an image on a per-instance basis. Amazon EC2 public images make use of this functionality to provide secure passwordless access to instances (and launching those images without a keypair ID will leave them inaccessible).

The public key material is made available to the instance at boot time by placing it in a file named `openssh_id.pub` on a logical device that is exposed to the instance as `/dev/sda2` (the ephemeral store). The format of this file is suitable for use as an entry within `~/.ssh/authorized_keys` (the OpenSSH format). This can be done at boot time (as part of `rclocal`, for example) allowing for secure password-less access. As the need arises, other formats will also be considered.

Request Parameters

The following table describes the request parameters for `RunInstances`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
`ImageId`	Id of the AMI to launch instances based on.	Yes	`string`

Element Name	Definition	Re-quired?	Type
MinCount	Minimum number of instances to launch.	Yes	int
MaxCount	Maximum number of instances to launch.	Yes	int
KeyName	Name of the keypair to launch instances with.	No	string
	Names of the security groups to asso-ciate the instances with.	No	string

Element Name	Definition	Required?	Type
n			
UserData	The user data available to the launched instances. This should be base64-encoded. See the UserData-Type data type for encoding details.	No	string

Response Tags

The following table describes the default response tags included in RunInstances responses.

Element Name	Definition
RunInstancesResponse	Status information about the instances started.

Sample Request

```
https://ec2.amazonaws.com/
?Action=RunInstances
&ImageId=ami-60a54009
&MaxCount=3
&MinCount=1
&...auth parameters...
```

Sample Response

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <reservationId>r-47a5402e</reservationId>
  <ownerId>495219933132</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-2ba64342</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
    </item>
    <item>
      <instanceId>i-2bc64242</instanceId>
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
    </item>
    <item>
      <instanceId>i-2be64332</instanceId>
```

```
      <imageId>ami-60a54009</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <dnsName></dnsName>
      <keyName>example-key-name</keyName>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

Related Operations

- DescribeInstances
- TerminateInstances
- AuthorizeSecurityGroupIngress
- RevokeSecurityGroupIngress
- DescribeSecurityGroups

TerminateInstances

The `TerminateInstances` operation shuts down one or more instances. This operation is idempotent and terminating an instance that is in the process of shutting down (or already terminated) will succeed.

Terminated instances remain visible for a short period of time (approximately one hour) after termination, after which their instance ID is invalidated.

Request Parameters

The following table describes the request parameters for `TerminateInstances`. Parameter names are case sensitive.

Element Name	Definition	Required?	Type
`InstanceId.n`	One or more instance IDs returned from previous calls to `RunInstances`.	Yes	`string`

Response Tags

The following table describes the default response tags included in `TerminateInstances` responses.

Element Name	Definition
`instancesSet`	A complex type containing describing the current and new state of each instance specified.

Sample Request

```
https://ec2.amazonaws.com/
?Action=TerminateInstances
&InstanceId.1=i-2ea64347
&InstanceId.2=i-21a64348
```

```
&...auth parameters...
```

Sample Response

```
<TerminateInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2007-01-03">
  <instancesSet>
    <item>
      <instanceId>i-28a64341</instanceId>
      <shutdownState>
        <code>32</code>
        <name>shutting-down</name>
      </shutdownState>
      <previousState>
        <code>0</code>
        <name>pending</name>
      </previousState>
    </item>
    <item>
      <instanceId>i-21a64348</instanceId>
      <shutdownState>
        <code>32</code>
        <name>shutting-down</name>
      </shutdownState>
      <previousState>
        <code>0</code>
        <name>pending</name>
      </previousState>
    </item>
  </instancesSet>
</TerminateInstancesResponse>
```

Related Operations

- DescribeInstances

Command Line Tools Reference

Introduction

The Amazon EC2 command line tools provide a command line interface to the web service API. This section describes each tool and its command line arguments in detail.

Command line options and arguments are based on the GNU getopt conventions. Optional parameters are indicated by means of flags. Flags typically come in a short and long form, although not all flags exist in both forms. In their short form, flags are a single character prefixed with a single dash. In their long form, flags use a longer, more expressive name prefixed with a double dash. Optional parameters typically have default values, or may be required only when other optional parameters are specified, and order is unimportant. For all remaining parameters order does matter.

A number of command line options apply to all of the command line tools. These are covered below and, for reasons of brevity, are not included in the description of each of the specific tools.

Errors

Any service errors encountered by the command line tools will be passed straight through from the API. A list of these errors can be seen in the section called “API Error Codes”.

Common Options

Most command line tools covered in the following sections accept a common set of optional parameters as follows:

Element Name	Definition	Valid Values/ Types	Example
-U URL	URL is the uniform resource locator of the Amazon EC2 web service entry point. This option defaults to the value of the `EC2_URL` environment variable, or http://ec2.amazonaws.com if that is not set.	URL	**`-U http://ec2.amazon-aws.com`**
-K EC2-PRIVATE-KEY	The private key to use when constructing requests to Amazon EC2. This parameter defaults to the value of the `EC2_PRIVATE_KEY` environment variable.	File name	**`-K pk-HKZYK-TAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem`**
-C EC2-CERT	The X509 certificate to use when constructing requests to Amazon EC2. This parameter defaults to the value of the `EC2_CERT` environment variable.	File name	**`-C cert-HKZYK-TAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem`**
-v	Increase output verbosity. This will print the SOAP request and response on the command line. This is particularly useful if you're trying to build your own tools to talk directly to our SOAP API.	N/A	N/A
--debug	Print internal debugging information. This is intended to assist us to troubleshoot problems.	N/A	N/A
-?	Show help.	N/A	N/A
-	If **`-`** is specified as an argument to one of the parameters, a list of arguments will be read from stdin. This is useful for piping the output of one command into the input of another.	N/A	**`ec2-describe-instances | grep running | cut -f 2 | ec2-terminate-instances -i -`**

By Function

AMI Tools

- ec2-bundle-image
- ec2-bundle-vol
- ec2-unbundle
- ec2-upload-bundle
- ec2-download-bundle
- ec2-delete-bundle

API Tools

Images

- ec2-register
- ec2-deregister
- ec2-describe-images

Instances

- ec2-run-instances
- ec2-describe-instances
- ec2-terminate-instances

Keypairs

- ec2-add-keypair
- ec2-describe-keypairs
- ec2-delete-keypair
- ec2-fingerprint-key

Image Attributes

- ec2-modify-image-attribute
- ec2-describe-image-attribute
- ec2-reset-image-attribute

Security Groups

- ec2-add-group

- ec2-delete-group
- ec2-describe-groups
- ec2-authorize
- ec2-revoke

ec2-add-group

SYNOPSIS

```
ec2-add-group GROUP -d DESCRIPTION
```

DESCRIPTION

Creates a new security group named GROUP. Group names must be unique per user.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("GROUP").
- Group name.
- Group description.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
-d DESCRIPTION	Description of the group. This is informational only.	Yes	-d 'Web servers'

EXAMPLE

```
$ ec2-add-group websrv -d 'Web servers'
GROUP websrv Web servers
```

SEE ALSO

- CreateSecurityGroup
- ec2-describe-groups
- ec2-delete-group
- ec2-authorize
- ec2-revoke

ec2-add-keypair

SYNOPSIS

```
ec2-add-keypair KEY
```

DESCRIPTION

A new 2048 bit RSA key pair is created with the specified name. The public key is stored by Amazon EC2 and the private key is displayed on the console. The private key is returned as an unencrypted PEM encoded PKCS#8 private key. If a key with the specified name already exists an error is returned.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("KEYPAIR").
- Keypair name.
- Private key fingerprint.
- Private key. This value is displayed on a new line.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-add-keypair gsg-keypair
KEYPAIR gsg-keypair
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQBuLFg5ujHrtm1jnutSuoO8Xe56LlT+HM8v/xkaa39EstM3/aFxTHgElQiJLChp
HungXQ29VTc8rc1bW0lkdi23OH5eqkMHGhvEwqa0HWASUMll4o3o/IX+0f2UcPoKCOVUR+jx71Sg
5AU52EQfanIn3ZQ8lFW7Edp5a3q4DhjGlUKToHVbicL5E+g45zfB95wIyywWZfeW/UUF3LpGZyq/
ebIUlq1qTbHkLbCC2r7RTn8vpQWp47BGVYGtGSBMpTRP5hnbzzuqj3itkiLHjU39S2sJCJ0TrJx5
i8BygR4s3mHKBj8l+ePQxG1kGbF6R4yg6sECmXn17MRQVXODNHZbAgMBAAECggEAY1tsiUsIwDl5
91CXirkYGuVfLyLflXenxfI50mDFms/mumTqloHO7tr0oriHDR5K7wMcY/YY5YkcXNo7mvUVD1pM
ZNUJs7rw9gZRTrf7LylaJ58kOcyajw8TsC4e4LPbFaHwS1d6K8rXh64o6WgW4SrsB6ICmr1kGQI7
3wcfgt5ecIu4TZf0OE9IHjn+2eRlsrjBdeORi7KiUNC/pAG23I6MdDOFEQRcCSigCj+4/mciFUSA
SWS4dMbrpb9FNSIcf9dcLxVM7/6KxgJNfZc9XWzUw77Jg8x92Zd0fVhHOux5IZC+UvSKWB4dyfcI
tE8C3p9bbU9VGyY5vLCAiIb4qQKBgQDLiO24GXrIkswF32YtBBMuVgLGCwU9h9HlO9mKAc2m8Cm1
jUE5IpzRjTedc9I2qiIMUTwtgnw42auSCzbUeYMURPtDqyQ7p6AjMujp9EPemcSVOK9vXYL0Ptco
xW9MC0dtV6iPkCN7gOqiZXPRKaFbWADp16p8UAIvS/a5XXk5jwKBgQCKkpHi2EISh1uRkhxljyWC
iDCiK6JBRsMvpLbc0v5dKwP5alo1fmdR5PJaV2qvZSj5CYNpMAy1/EDNTY5OSIJU+0KFmQbyhsbm
rdLNLDL4+TcnT7c62/aH01ohYaf/VCbRhtLlBfqGoQc7+sAc8vmKkesnF7CqCEKDyF/dhrxYdQKB
gC0iZzzNAapayz1+JcVTwwEid6j9JqNXbBc+Z2YwMi+T0Fv/P/hwkX/ypeOXnIUcw0Ih/YtGBVAC
DQbsz7LcY1HqXiHKYNWNvXgwwO+oiChjxvEkSdsTTIfnK4VSCvU9BxDbQHjdiNDJbL6oar92UN7V
rBYvChJZF7LvUH4YmVpHAoGAbZ2X7XvoeEO+uZ58/BGKOIGHByHBDiXtzMhdJr15HTYjxK7OgTZm
gK+8zp4L9IbvLGDMJO8vft32XPEWuvI8twCzFH+CsWLQADZMZKSsBasOZ/h1FwhdMgCMcY+Qlzd4
JZKjTSu3i7vhvx6RzdSedXEMNTZWN4qlIx3kR5aHcukCgYA9T+Zrvm1F0seQPbLknn7EqhXIjBaT
P8TTvW/6bdPi23ExzxZn7KOdrfclYRph1LHMpAONv/x2xALIf91UB+v5ohy1oDoasL0gij1houRe
2ERKKdwz0ZL9SWq6VTdhr/5G994CK72fy5WhyERbDjUIdHaK3M849JJuf8cSrvSb4g==
-----END RSA PRIVATE KEY-----
```

SEE ALSO

- CreateKeypair

- ec2-describe-keypairs
- ec2-delete-keypair

ec2-authorize

SYNOPSIS

```
ec2-authorize GROUP [-P PROTOCOL] (-p PORT_RANGE | -t ICMP_TYPE_CODE) [-u
SOURCE_GROUP_USER ...] [-o SOURCE_GROUP ...] [-s SOURCE_SUBNET ...]
```

DESCRIPTION

Adds a rule to the security group named GROUP. If no source host, group or subnet is provided, requests from any source address will be honored.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("GROUP", "PERMISSION").
- Group name. Currently, this will report an empty string.
- Type of rule. Currently, only ALLOW rules are supported.
- Protocol to allow.
- Start of port range.
- End of port range.
- `FROM`
- Source.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
`-P PROTOCOL`	The protocol to allow. This can be `tcp`, `udp` or `icmp`. This option only applies when specifying a CIDR subnet as the source.	Yes	-P tcp
`-p PORT_RANGE`	For the TCP or UDP protocols, this specifies the range of ports to allow. This may be specified as a single integer or as a range (min-max). This option only applies when specifying a CIDR subnet as the source.	Yes	-p 80
`-t ICMP_TYPE_CODE`	For the ICMP protocol, the ICMP type and code must be specified. This must be specified as type:code where both are integers. Type or code (or both) may be specified as -1 which is a wildcard. This option only applies when specifying a CIDR subnet as the	Yes	-t 2:5

Option	Definition	Required?	Example
	source.		
-u SOURCE_GROUP_USER	The owner of a group specified using *-o*. If this is not specified, all groups will refer to the current user. If specified more than once, there must be exactly one *-u* per *-o* and each user will be mapped to the corresponding group.	No	-u 495219933132
-o SOURCE_GROUP	The network source from which traffic is to be authorized specified as a security Group. See the description of the *-u* parameter for group owner information.	No	-o headoffice
-s SOURCE_SUBNET	The network source from which traffic is to be authorized specified as a CIDR Subnet range.	No	-s 205.192.8.45/24

EXAMPLE

```
$ ec2-authorize websrv -P tcp -p 80 -s 205.192.0.0/16
GROUP websrv ""
PERMISSION websrv ALLOWS tcp 80 80 FROM CIDR 205.192.0.0/16
```

SEE ALSO

- AuthorizeSecurityGroupIngress
- ec2-add-group
- ec2-describe-groups
- ec2-delete-group
- ec2-revoke

ec2-bundle-image

SYNOPSIS

```
ec2-bundle-image -k PRIVATE-KEY -u USER-ID -i IMAGE [-d DESTINATION-DIR] [-p
AMI-PREFIX]
```

DESCRIPTION

Create a bundled AMI of an operating system image that was created in a loopback file.

OUTPUT

Status messages indicating the various stages of the bundling process are displayed.

OPTIONS

Note
Note that this tool does not support the common options

Option	Definition	Required?	Example
-k, --privatekey KEY	The path to the user's PEM encoded RSA key file.	Yes	-k $HOME/pk-234242DEADCAFE.pem
-u, --user USER	The user's EC2 user ID (a.k.a. AWS account number).	Yes	-u 123456789
-i, --image PATH	The path to the image to bundle.	Yes	-i / var/ spool/ my-image/version-2/debian.img
-d, --destination DESTINATION	The directory in which to create the bundle. Defaults to the current directory.	No	-d /var/run/my-bundle
-p, --prefix PREFIX	The filename prefix for bundled AMI files. Defaults to "image".	No	-p my-image-is-special
--help	Display the help message.	No	--help
--manual	Display the help.	No	--manual

EXAMPLE

```
$ ec2-bundle-image -k ./my-very-secret-key/pk-very-very-secret-key.pem -u
DEAD0CAFE1AND3BEEF -i image.img -d bundled/ -p fred
Splitting bundled/fred.gz.crypt...
Created fred.part.00
Created fred.part.01
Created fred.part.02
Created fred.part.03
```

```
Created fred.part.04
Created fred.part.05
Created fred.part.06
Created fred.part.07
Created fred.part.08
Created fred.part.09
Created fred.part.10
Created fred.part.11
Created fred.part.12
Created fred.part.13
Created fred.part.14
Generating digests for each part...
Digests generated.
Creating bundle manifest...
Bundle Image complete.
```

SEE ALSO

- ec2-bundle-vol
- ec2-unbundle
- ec2-upload-bundle
- ec2-download-bundle
- ec2-delete-bundle

ec2-bundle-vol

SYNOPSIS

```
ec2-bundle-vol -k PRIVATE-KEY-u USER-ID -s SIZE [-d DESTINATION-DIR] [-e
EXLCUDE-DIR-1,EXCLUDE-DIR-2...] [-p AMI-PREFIX] [-v VOLUME]
```

DESCRIPTION

Create a bundled AMI by taking a snapshot of the local machine's root file system, compressing, encrypting and signing the snapshot.

OUTPUT

Status messages indicating the various stages of the bundling process are displayed.

OPTIONS

Note

Note that this tool does not support the common arguments

Option	Definition	Required?	Example
-k, --privatekey KEY	The path to the user's PEM encoded RSA key file.	Yes	-k $HOME/pk-234242DEADCAFE.pem
-u, --user USER	The user's EC2 user ID (a.k.a. AWS account number).	Yes	-u 123456789
-s, --size SIZE	The size, in MB (1024 * 1024 bytes), of the image file to create. The maximum size is 10240 MB.	Yes	-s 2048
-d, --destination DESTINATION	The directory in which to create the bundle. Defaults to "/tmp".	No	-d /var/run/my-bundle
-e, --exclude DIR1,DIR2,...	A list of absolute directory paths to exclude from the bundle operation. Note that it overrides the '--all' parameter.	No	-e /tmp,/home/secret-data
-p, --prefix PREFIX	The filename prefix for bundled AMI files. Defaults to "image".	No	-p my-image-is-special
-v, --volume VOLUME	The absolute path to the mounted volume to create the bundle from. Defaults to "/".	No	-v /mnt/my-customized-ami
-a, --all	Bundle all directories, including those on remotely mounted filesystems.	No	-a
--help	Display the help message.	No	--help
--manual	Display the user manual.	No	--manual

EXAMPLE

```
$ ec2-bundle-vol -d /mnt -k ~root/pk-HKZYKTAIG2ECMXYIBH3HXV4ZBZQ55CLO.pem -u
495219933132 -s 1536
Copying / into the image file /mnt/image.img...
Excluding:
     sys
     dev/shm
     proc
     dev/pts
     proc/sys/fs/binfmt_misc
     dev
     media
     mnt
     proc
     sys
     tmp/image.img
     mnt/img-mnt
1+0 records in
1+0 records out
mke2fs 1.38 (30-Jun-2005)
warning: 256 blocks unused.

Splitting /mnt/image.gz.crypt...
Created image.part.00
Created image.part.01
Created image.part.02
Created image.part.03
...
Created image.part.22
Created image.part.23
Generating digests for each part...
Digests generated.
Creating bundle manifest...
Bundle Volume complete.
```

SEE ALSO

- ec2-bundle-image
- ec2-unbundle
- ec2-upload-bundle
- ec2-download-bundle
- ec2-delete-bundle

ec2-delete-bundle

SYNOPSIS

```
ec2-delete-bundle -b S3-BUCKET -a AWS-ACCESS-KEY-ID -s AWS-SECRET-KEY [-m
MANIFEST-PATH] [-p PREFIX] [--url URL] [--retry] [-y]
```

DESCRIPTION

Delete the specified bundle from S3 storage.

OUTPUT

Status messages indicating the various stages of the delete process are displayed.

OPTIONS

Note

Note that this tool does not support the common arguments

Option	Definition	Required?	Example
-b, --bucket S3-BUCKET	The name of the Amazon S3 bucket containing the bundled AMI	Yes	-b aes-crack-er-ami-bucket
-a, --access-key USER	The user's AWS access key ID.	Yes	-a ???????
-s, --secret-key PASSWORD	The user's AWS secret access key.	Yes	-s ???????
-m, --manifest MANIFEST-PATH	The path to the unencrypted manifest file.	Yes	-m / var/ spool/ my-first-bundle/Manifest
-p, --prefix PREFIX	The bundled AMI part filename prefix.	No	-p eos-
--url URL	The S3 service URL. Defaults to ht-tps://s3.amazonaws.com.	No	--url ht-tps://s3.amazonaws.ie
--retry	Automatically retry failed uploads. Use with caution.	No	--retry
-y, --yes	Automatically assume the answer to all prompts is 'yes'.	No	-y
--help	Display the help message.	No	--help
--manual	Display the help.	No	--manual

EXAMPLE

```
$ ec2-delete-bundle -b my-s3-bucket -a ??????????????????????? -s
???????????????????????????????? -p fred


??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????
```

SEE ALSO

- ec2-bundle-image
- ec2-bundle-vol
- ec2-unbundle
- ec2-upload-bundle
- ec2-download-bundle

ec2-delete-group

SYNOPSIS

```
ec2-delete-group GROUP
```

DESCRIPTION

Deletes the named GROUP.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("GROUP").
- Name of the deleted group.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-delete-group websrv
GROUP websrv
```

SEE ALSO

- DeleteSecurityGroup
- ec2-add-group
- ec2-describe-groups
- ec2-authorize
- ec2-revoke

ec2-delete-keypair

SYNOPSIS

```
ec2-delete-keypair KEY
```

DESCRIPTION

Deletes the named KEY, purging the public key from Amazon EC2

OUTPUT

A table containing the following information is returned:

- Output type identifier ("KEYPAIR").
- Identifier of the deleted keypair.
- Private key fingerprint.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-delete-keypair gsg-keypair
KEYPAIR gsg-keypair
```

SEE ALSO

- DeleteKeypair
- ec2-add-keypair
- ec2-describe-keypairs

ec2-deregister

SYNOPSIS

```
ec2-deregister AMI
```

DESCRIPTION

The AMI identified is deregistered. This AMI may no longer be used to launch new instances. The AMI is not deleted from Amazon S3

OUTPUT

A table containing the following information is returned:

- A record type identifier ("IMAGE")
- the image identifier that was deregistered

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-deregister ami-4fa54026
IMAGE ami-4fa54026
```

SEE ALSO

- DeregisterImage
- ec2-register
- ec2-describe-images

ec2-describe-groups

SYNOPSIS

```
ec2-describe-groups [GROUP ...]
```

DESCRIPTION

Describes the current state of each GROUP specified on the command line. If no GROUPs are explicitly listed then all GROUPs owned by the current user are included in the output.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("GROUP", "PERMISSION").
- User ID of group owner.
- Group name.
- Description of the group.
- Firewall rule.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-describe-groups websrv
GROUP 495219933132 websrv Web servers
PERMISSION 495219933132 websrv ALLOWS tcp 80 80 FROM CIDR 0.0.0.0/0
```

SEE ALSO

- DescribeSecurityGroups
- ec2-add-group
- ec2-delete-group
- ec2-authorize
- ec2-revoke

ec2-describe-image-attribute

SYNOPSIS

```
ec2-describe-image-attribute AMI -l
```

DESCRIPTION

Describes an attribute for the specified AMI.

OUTPUT

A table containing the following information is returned:

- Attribute type identifier
- ID of the AMI of which an attribute is being described.
- Attribute value type or attribute list item value type.
- Attribute or attribute list item value.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
-l	Describes the `launchPermission` attribute.	Yes	-l

EXAMPLE

```
$ ec2-describe-image-attribute ami-5bae4b32 -l
launchPermission ami-5bae4b32 group all
launchPermission ami-5bae4b32 userId 495219933132
```

SEE ALSO

- DescribeImageAttribute
- ec2-modify-image-attribute
- ec2-reset-image-attribute
- Sharing AMIs

ec2-describe-images

SYNOPSIS

```
ec2-describe-images[AMI ...] [-a] [-o OWNER ...] [-x USER]
```

DESCRIPTION

Describes the current state of each AMI specified on the command line. If no AMIs are explicitly listed then the AMIs described can be controlled with the optional parameters. If no optional parameters are specified all AMIs that the user owns or has explicit launch permissions for are displayed.

Note

The default behaviour of `ec2-describe-images` has changed from version 2006-06-26 to version 2006-10-01. In the 2006-06-26 version all images the user has access to, including public images, are returned. In the 2006-10-01 version only images the user owns or has explicit access to are returned. Public images are not returned.

OUTPUT

A table containing the following information is returned:

- A record type identifier ("IMAGE")
- image identifier
- manifest location
- user identifier of the user that registered the image
- image status
- `public` or `private` indicating whether or not the image is visible to all users

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
-a	All AMIs the user owns and has execution permissions for, both public and explicit are returned.	No	-a
-o OWNER	AMIs owned by the specified owner are returned. Multiple owners may be specified. OWNER is an AWS account id or one of the following special values: • *amazon* Include AMIs owned by Amazon in the result set. • *self* Include AMIs owned by the caller in the result set.	No	-o 123456789012
-x USER	Returns AMIs owned by the caller	No	-x self

Option	Definition	Required?	Example
	which the specified user has launch permissions for. Multiple users may be specified. USER is an AWS account id or one of the following special values: • *all* Include public AMIs in the result set. • *self* Include AMIs the caller has explicit launch permissions for in the result set.		

EXAMPLE

```
$ ec2-describe-images ami-78a54011
IMAGE ami-78a54011 powerdns/image.manifest.xml 495219933132 available private
```

SEE ALSO

- DescribeImages
- ec2-register
- ec2-deregister

ec2-describe-instances

SYNOPSIS

```
ec2-describe-instances [INSTANCEID ...]
```

DESCRIPTION

Describes the current state of each instance indicated by the respective INTANCEID specified on the command line. If no instances are explicitly listed then all instances owned by the current user are included in the output.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("RESERVATION", "INSTANCE")
- Instance ID which uniquely identifies each running instance.
- AMI ID of the image the instance is based on.
- DNS name associated with the instance (only present for instances in the `running` state).
- Instance state.
- Key name. If a key was associated with the instance at launch it's name will be displayed in this column.
- AMI launch index. See using instance data for more info.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-describe-instances
RESERVATION r-15a4417c 495219933132
INSTANCE i-3ea74257 ami-6ba54002 domU-
12-31-33-00-00-01.dc3.compute.amazonaws.com running 0
INSTANCE i-31a74258 ami-6ba54002 domU-
12-31-33-00-00-02.dc3.compute.amazonaws.com running 1
```

SEE ALSO

- DescribeInstances
- ec2-run-instances
- ec2-terminate-instances

ec2-describe-keypairs

SYNOPSIS

```
ec2-describe-keypairs [KEY ...]
```

DESCRIPTION

Describes the current state of each KEY specified on the command line. If no KEYs are explicitly listed then all KEYs owned by the current user are included in the output.

OUTPUT

A table containing the following information is returned:

- A output type identifier ("KEYPAIR")
- Keypair identifier
- Private key fingerprint

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-describe-keypairs gsg-keypair
KEYPAIR gsg-keypair
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
```

SEE ALSO

- DescribeKeypairs
- ec2-add-keypair
- ec2-delete-keypair

ec2-download-bundle

SYNOPSIS

```
ec2-download-bundle -b S3-BUCKET -m MANIFEST -a AWS-ACCESS-KEY-ID -s
AWS-SECRET-KEY -k PRIVATE-KEY [-d DIRECTORY] [--url URL]
```

DESCRIPTION

Download the specified bundles from S3 storage.

OUTPUT

Status messages indicating the various stages of the download process are displayed.

OPTIONS

Note

Note that this tool does not support the common arguments

Option	Definition	Required?	Example
-b, --bucket S3-BUCKET	The name of the Amazon S3 bucket from which to fetch the bundles.	Yes	-b aes-cracked
-m, --manifest MANIFEST	The manifest filename.	Yes	-m /var/spool/my-first-bundle/Manifest
-a, --access-key USER	The user's AWS access key ID.	Yes	-a ???????
-s, --secret-key PASSWORD	The user's AWS secret access key.	Yes	-s ???????
-k, --privatekey KEY	The user's private key used to decrypt the manifest.	Yes	-k ???????
-d, --directory DIRECTORY	The directory into which the downloaded bundles are saved. Defaults to the current working directory. **Note** The directory must exist.	No	-d /tmp/my-downloaded-bundle
--url URL	The S3 service URL. Defaults to https://s3.amazonaws.com.	No	--url https://s3.amazonaws.ie
--help	Display the help message.	No	--help

EXAMPLE

```
$ mkdir unbundled
$ ec2-download-bundle -b my-s3-bucket -m fred.manifest.xml -a
??????????????????????? -s ????????????????????????????????? -k
./my-very-secret-key/pk-very-very-secret-key.pem -d bundled


??????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????
```

SEE ALSO

- ec2-bundle-image
- ec2-bundle-vol
- ec2-unbundle
- ec2-upload-bundle
- ec2-delete-bundle

ec2-fingerprint-key

SYNOPSIS

```
ec2-fingerprint-key KEYFILE
```

DESCRIPTION

Computes and displays the fingerprint for a private key produced by Amazon EC2. KEYFILE must be the path to a file containing an unencrypted PEM encoded PKCS#8 private key.

This operation is performed entirely on the client-side. Network access is not required.

OUTPUT

A key fingerprint. This is formatted as a hash digest with each octet separated by a colon.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-fingerprint-key mykey.pem
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f
```

SEE ALSO

- ec2-create-keypair
- ec2-describe-keypairs

ec2-get-console-output

SYNOPSIS

```
ec2-get-console-output INSTANCEID [-r]
```

DESCRIPTION

Retrieve the console output for instance INSTANCEID, if available, and display it to `stdout`.

OUTPUT

Two fields:

- A timestamp indicating the time of the last update.
- The instance console output. By default the `^ESC` character is escaped and duplicate new-lines are removed to facilitate reading.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
-r	Raw output. Do not escape the output to facilitate reading.	No	

EXAMPLE

```
$ ec2-get-console-output  i-10a64379
2007-01-03 12:00:00
Linux version 2.6.16-xenU (builder@patchbat.amazonsa) (gcc version 4.0.1
20050727 (Red Hat 4.0.1-5)) #1 SMP Thu Oct 26 08:41:26 SAST 2006
BIOS-provided physical RAM map:
Xen: 0000000000000000 - 000000006a400000 (usable)
980MB HIGHMEM available.
727MB LOWMEM available.
NX (Execute Disable) protection: active
IRQ lockup detection disabled
Built 1 zonelists
Kernel command line: root=/dev/sda1 ro 4
Enabling fast FPU save and restore... done.
```

ec2-modify-image-attribute

SYNOPSIS

```
ec2-modify-image-attribute AMI -l (-a ITEM_VALUE | -r ITEM_VALUE)
```

DESCRIPTION

Modifies an attribute for the specified AMI.

Currently the only attribute supported is launchPermission. By modifying this attribute it is possible to make an AMI public or to grant specific users launch permissions for the AMI. To make the AMI public add `all`. To grant launch permissions for a specific user add that user's AWS account id.

Note
If another user launches your AMI there is nothing that prevents that user from rebundling the image and registering it as a new AMI.

OUTPUT

A table containing the following information is returned:

- Attribute type identifier.
- ID of the AMI on which attributes are being modified.
- Action performed on the attribute.
- Attribute or attribute list item value type.
- Attribute or attribute list item value.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
-l	Modify the launchPermission property.	Yes	-l
-a ITEM_VALUE | -r ITEM_VALUE	Adds or removes an attribute item. The value of the item is `ITEM_VALUE`. The type of the item is inferred from the item value. For launchPermission there are two item types: • `group`: The only group currently supported is the `all` group. Adding this group sets public launch permissions for the AMI. • `userId`: UserId must be in the form of an AWS account id. Adding userId items grants explicit launch permissions to that user for the AMI.	Yes	-a all

Option	Definition	Required?	Example

EXAMPLE

```
$ ec2-modify-image-attribute ami-5bae4b32 -l -a 495219933132
launchPermission ami-5bae4b32 ADD userId 495219933132
```

SEE ALSO

- ModifyImageAttribute
- ec2-reset-image-attribute
- ec2-describe-image-attribute
- Sharing AMIs

ec2-reboot-instances

SYNOPSIS

```
ec2-reboot-instances INSTANCEID [INSTANCEID ...]
```

DESCRIPTION

All instances indicated by the respective INTANCEID specified on the command line are rebootd. At least one INSTANCEID must be specified.

OUTPUT

This command displays no output on success.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-reboot-instances i-3ea74257
```

ec2-register

SYNOPSIS

```
ec2-register MANIFEST
```

DESCRIPTION

Registers the Amazon Machine Image (AMI) described by the named MANIFEST file, generating a new Amazon Machine Image (AMI) ID. MANIFEST must specify a location of a manifest file in Amazon S3 and must be of the form `bucket/object`.

OUTPUT

The image ID that was assigned by Amazon EC2 is displayed.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-register mybucket/image.manifest.xml
IMAGE ami-78a54011
```

SEE ALSO

- RegisterImage
- ec2-deregister
- ec2-describe-images

ec2-reset-image-attribute

SYNOPSIS

```
ec2-reset-image-attribute AMI -l
```

DESCRIPTION

Resets an attribute for the specified AMI.

OUTPUT

A table containing the following information is returned:

- Attribute type identifier
- ID of the AMI on which the attribute is being reset
- Action identifier ("RESET")

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
-l	Reset the `launchPermission` attribute.	Yes	-l

EXAMPLE

```
$ ec2-reset-image-attribute ami-6ba54002 -l
launchPermission ami-6ba54002 RESET
```

SEE ALSO

- ResetImageAttribute
- ec2-modify-image-attribute
- ec2-describe-image-attribute
- Sharing AMIs

ec2-revoke

SYNOPSIS

```
ec2-revoke GROUP [-P PROTOCOL] (-p PORT_RANGE | -t ICMP_TYPE_CODE) [-u
SOURCE_GROUP_USER ...] [-o SOURCE_GROUP ...] [-s SOURCE_SUBNET ...]
```

DESCRIPTION

Revokes a rule from the security group named GROUP. To identify the rule to be removed you must provide exactly the same set of options used to create that rule.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("GROUP", "PERMISSION").
- Group name. Currently, this will report an empty string.
- Type of rule. Currently, only ALLOW rules are supported.
- Protocol to allow.
- Start of port range.
- End of port range.
- `FROM`
- Source.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
-P PROTOCOL	The protocol to allow. This can be`tcp`, `udp` or `icmp`. This option only applies when specifying a CIDR subnet as the source.	Yes	-P tcp
-p PORT_RANGE	The range of ports to revoke. This may be specified as a single integer or as a range (min-max). This option only applies when specifying a CIDR subnet as the source.	Yes	-p 80
-t ICMP_TYPE_CODE	If the protocol is ICMP, the ICMP type and code must be specified. This must be specified as type:code where both are integers. Type or code (or both) may be specified as -1 which acts as a wildcard. This option only applies when specifying a CIDR subnet as the source.	Yes	-t 2:5

Option	Definition	Required?	Example
-u SOURCE_GROUP_USER	The owner of a group specified using *-o*. If this is not specified, all groups will refer to the current user. If specified more than once, there must be exactly one *-u* per *-o* and each user will be mapped to the corresponding group.	No	-u 495219933132
-o SOURCE_GROUP	The network source from which traffic is to be revoked specified as a security Group. See the description of the *-u* parameter for group owner information.	No	-o outsideworld
-s SOURCE_SUBNET	The network source from which traffic is to be revoked specified as a CIDR Subnet range.	No	-s 205.192.8.45/24

EXAMPLE

```
$ ec2-revoke websrv -P tcp -p 80 -s 205.192.0.0/16
GROUP websrv ""
PERMISSION websrv ALLOWS tcp 80 80 FROM CIDR 205.192.0.0/16
```

SEE ALSO

- RevokeSecurityGroupIngress
- ec2-add-group
- ec2-describe-groups
- ec2-delete-group
- ec2-authorize

ec2-run-instances

SYNOPSIS

```
ec2-run-instances AMI [-n INSTANCE_COUNT] [-g GROUP [-g GROUP ...]] [-k KEY]
[-d USER_DATA | -f FILE_NAME]
```

DESCRIPTION

Launches one or more instances of the specified AMI. Optional parameters include

- A security group. New instances will be launched in this group. If no group is specified instances are launched in the `default` group.
- A keypair name. The public key associated with this keypair name will be made available to the instances at boot time.
- User data. This data will be made available to the launched instances. See using instance data for more info.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("INSTANCE")
- Instance ID which uniquely identifies each running instance.
- AMI ID of the image the instance is based on.
- DNS name associated with the instance (only present for instances in the `running` state).
- Instance state. This will in most cases be *`pending`* which indicates that the instance is being prepared for launch.
- Key name. If a key was associated with the instance at launch it's name will be displayed in this column.

Errors are displayed on stderr.

OPTIONS

Option	Definition	Required?	Example
`-n IN-STANCE_COUNT`	The number of instances to launch. If not specified, a value of `1` will be assumed. If it is not possible to launch at least this many instances (due to a lack of capacity or funds), no instances will be launched. If specified as a range (min-max) Amazon EC2 will try to launch as many instances as possible, up to max, but will launch no fewer than min instances.	No	-n 5
`-g GROUP`	The security group(s) within which the instance(s) should be run. This determ-	No	-g fooGroup

Option	Definition	Required?	Example
	ines the ingress firewall rules that will be applied to the instances. By default instances will run in the user's default group. If more than one group is specified, the security policy of the instances will be the union of the security policies of the specified groups.		
-k KEY	The keypair to make available to these instances at boot time.	No	-k fooKeyPair
-d USER-DATA	The data to make available to these instances. The data is read off the command line from the `USER_DATA` argument. If you want the data to be read from a file see the `-f` option.	No	-d "my user data"
-f FILE_NAME	The data to make available to these instances. The data is read from the file specified by `FILE_NAME`. If you want to specify user data on the command line use the `-d` flag. `-d` option.	No	-f data.zip

EXAMPLE

```
$ ec2-run-instances ami-6ba54002 -n 5
INSTANCE i-3ea74257 ami-6ba54002 pending
INSTANCE i-31a74258 ami-6ba54002 pending
INSTANCE i-31a74259 ami-6ba54002 pending
INSTANCE i-31a7425a ami-6ba54002 pending
INSTANCE i-31a7425b ami-6ba54002 pending
INSTANCE i-31a7425c ami-6ba54002 pending
```

SEE ALSO

- RunInstances
- ec2-terminate-instances
- ec2-describe-instances
- ec2-add-keypair
- Using instance data

ec2-terminate-instances

SYNOPSIS

```
ec2-terminate-instances INSTANCEID [INSTANCEID ...]
```

DESCRIPTION

All instances indicated by the respective INTANCEID specified on the command line are terminated. At least one INSTANCEID must be specified.

OUTPUT

A table containing the following information is returned:

- Output type identifier ("INSTANCE")
- The instance ID of the instance being terminated.
- The state of the instance prior to being terminated.
- The new state of the instance.

Errors are displayed on stderr.

EXAMPLE

```
$ ec2-terminate-instances i-3ea74257
INSTANCE i-3ea74257 running shutting-down
```

SEE ALSO

- TerminateInstances
- ec2-run-instances
- ec2-describe-instances

ec2-unbundle

SYNOPSIS

```
ec2-unbundle -m MANIFEST-PATH [-d DESTINATION-DIRECTORY] [-s
SOURCE-DIRECTORY]
```

DESCRIPTION

Recreates the AMI from the bundled AMI parts.

OUTPUT

Status messages indicating the various stages of the unbundling process are displayed.

OPTIONS

Note

Note that this tool does not support the common arguments

Option	Definition	Required?	Example
-m, --manifest MANIFEST	The path to the unencrypted AMI manifest file.	Yes	-m /var/spool/my-first-bundle/Manifest
-s, --source SOURCE-DIRECTORY	The directory containing the bundled AMI parts. Defaults to the current directory.	No	-s /tmp/my-bundled-image
-d, --destination DESTINATION-DIRECTORY	The directory to unbundle the AMI in. Defaults to the current directory. **Note** The destination directory must exist.	No	-d /tmp/my-image
--help	Display the help message.	No	--help

EXAMPLE

```
$ mkdir unbundled
$ ec2-unbundle -m fred.manifest.xml -s bundled -d unbundled
cat  bundled/fred.part.00 bundled/fred.part.01 bundled/fred.part.02 bundled/
fred.part.03 bundled/fred.part.04 bundled/fred.part.05 bundled/fred.part.06
bundled/fred.part.07 bundled/fred.part.08 bundled/fred.part.09 bundled/
fred.part.10 bundled/fred.part.11 bundled/fred.part.12 bundled/fred.part.13
bundled/fred.part.14 | openssl enc -d -aes-128-cbc -K
a8fbe9586b7fd3df893b237f88e351a9 -iv 121febdf64b0322cd4ffda03aa1ab535 | gun-
zip > unbundled/fred.img
Unbundle complete.
$ ls -l unbundled
total 1025008
```

```
-rw-r--r--  1 root root 1048578048 Aug 25 23:46 fred.img
```

SEE ALSO

- ec2-bundle-image
- ec2-bundle-vol
- ec2-upload-bundle
- ec2-download-bundle
- ec2-delete-bundle

ec2-upload-bundle

SYNOPSIS

```
ec2-upload-bundle -b S3-BUCKET -m MANIFEST-PATH -a AWS-ACCESS-KEY-ID -s
AWS-SECRET-KEY [--acl ACL] [--ec2certificate PATH] [-d DIRECTORY] [--part
PART] [--url URL] [--retry] [--skipmanifest]
```

DESCRIPTION

Upload a bundled AMI to S3 storage.

OUTPUT

Status messages indicating the various stages of the upload process are displayed.

OPTIONS

Note

Note that this tool does not support the common options

Option	Definition	Required?	Example
-b, --bucket S3-BUCKET	The name of the Amazon S3 bucket in which the bundle will be stored. If the bucket doesn't exist it will be created (provided the bucket is available of course).	Yes	-b aes-cracker-ami
-m, --manifest MANIFEST-PATH	The path to the manifest file. The manifest file is created during the bundling process and can be found in the directory containing the bundle.	Yes	-m /var/spool/my-first-bundle/Manifest
-a, --access-key USER	The user's AWS access key ID.	Yes	-a ???????
-s, --secret-key PASSWORD	The user's AWS secret access key.	Yes	-s ???????
--acl ACL	The access control list policy of the bundled image. It may be either "public-read" or "aws-exec-read" and defaults to "aws-exec-read" if not specified.	No	--acl public-read
--ec2certificate PATH	The path to the EC2 X509 public key certificate. Defaults to "/etc/aes/amiutil/cert-ec2.pem".	No	--ec2certificate $HOME/pk-234242DEADCAFE.pem
-d, --directory	The directory containing the bundled AMI parts. Defaults to the directory	No	-d /var/run/my-bundle

Option	Definition	Required?	Example
DIRECTORY	containing the manifest file (see the "-m" option).		
--part PART	Start uploading the specified part and upload all subsequent parts.	No	--part ????
--url URL	The S3 service URL. Defaults to https://s3.amazonaws.com.	No	--url https://s3.amazonaws.ie
--retry	Automatically retry failed uploads. Use with caution.	No	--retry
--skipmanifest	Do not upload the manifest.	No	--skipmanifest
--help	Display the help message.	No	--help
--manual	Display the help.	No	--manual

EXAMPLE

```
$ ec2-upload-bundle -b my-s3-bucket -m bundled/fred.manifest.xml -u
DEAD0CAFE1AND3BEEF -p mySecretPasswordGoesHereButThisIsntIt03m -d bundled
Encrypting bundle manifest...
Completed encryption.
Uploading encrypted manifest...
Uploaded encrypted manifest to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.manifest.xml.
Uploading bundled AMI parts to http://s3.amazonaws.com:80/alpowell-images...
Uploaded fred.part.00 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.00.
Uploaded fred.part.01 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.01.
Uploaded fred.part.02 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.02.
Uploaded fred.part.03 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.03.
Uploaded fred.part.04 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.04.
Uploaded fred.part.05 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.05.
Uploaded fred.part.06 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.06.
Uploaded fred.part.07 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.07.
Uploaded fred.part.08 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.08.
Uploaded fred.part.09 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.09.
Uploaded fred.part.10 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.10.
Uploaded fred.part.11 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.11.
Uploaded fred.part.12 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.12.
Uploaded fred.part.13 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.13.
Uploaded fred.part.14 to ht-
tp://s3.amazonaws.com:80/alpowell-images/fred.part.14.
Upload Bundle complete.
```

SEE ALSO

- ec2-bundle-image
- ec2-bundle-vol
- ec2-unbundle
- ec2-download-bundle
- ec2-delete-bundle

Technical FAQ

1. Why can't I "talk" to my instances?

   Here are a few common reasons for broken connectivity to your instance.

   An instance's state is changed to running as soon as we start to boot your OS. This means there will be some delay (possibly a few minutes depending on your configuration) during which your instance will not have been fully set-up. After this period, it should be fully functional.

   Additionally, you will need to make sure you have authorized the appropriate access to your host through the Amazon EC2 firewall. If you have launched your instances without specifying a security group, the `default` group is used. Permissions on the `default` group are very strict by default and disallow all access from the Internet and other groups. You will need to add permissions to your `default` group or you will have to set up a new group with appropriate permissions. See the developer guide for more information on the “Securing the Network”.

   Assuming you have authorized port 22, a useful debugging tool is to try to open an ssh connection with verbose output. You should use the man page to get the exact syntax for your system, but the command is likely to look like `ssh -vv root@[hostname]`. This output would be very useful if posting to the forum.

2. Why did my instance terminate immediately after launch?

   Launch errors may be the result of an internal error during launch or a corrupt Amazon EC2 image. The former should be rare, and we actively test for and isolate suspect hosts. You should use the “DescribeInstances” API to look for more details on why your instance failed to launch.

   NB: the `ec2-describe-instances` command line tool does not conveniently print out this information yet! You can use the `-v` flag to read the SOAP response from this tool and get the information discussed above.

   You can always feel free to attempt to launch the image again, but if you run into a persistent problem (especially with a shared image), you should post to the Amazon EC2 forum.

3. I ran `shutdown` from within an `ssh` session but my instance still shows up as running when I query it with DescribeInstances and I can't shell into it. What's happening?

   This is a "feature" of the `shutdown` command. If you issue `shutdown` without a `-h` (halt) flag it shuts down the network and switches to single user mode. The instance is still running but without a network. You should always use `shutdown -h` when working inside an Amazon EC2 instance.

   You can shut the instance down using the TerminateInstances call (`ec2-terminate` on the command line).

4. What username do I use for the various Amazon EC2 tools?

   When you sign up with Amazon Web Services, you are given an AWS Account ID. This is your username. More detail is provided in the Getting Started Guide.

5. What happens to my running instances if the machines they are running on go down?

   The instances themselves will be terminated and will have to be relaunched. The data on the instances' hard drives will be lost.

Always replicate important data or store it in Amazon S3.

6. Why are my instances stuck in a pending state (or a shutting-down state)?

   This situation should be rare and is the result of a software error or misconfiguration. We actively monitor for it, but please let us know if you do encounter this.

7. Why do I get an "AuthFailure: User is not AMI creator" error when I try to register an image?

   Make sure that you are using the correct user ID and certificate to create and upload the image. You need to use the same ID and certificate to register the image with Amazon EC2.

8. Why do I get an "InsufficientInstanceCapacity" error when I try to launch an instance?

   This error indicates that we don't currently have enough available capacity to service your request. During our beta, capacity is limited.

   If you are requesting a large number of instances, there may not be enough server capacity to host them. You could try again at a different time or specify a smaller number of instances to launch.

9. Why do I get an "InstanceLimitExceeded" error when I try to launch an instance?

   This error indicates that you have reached your concurrent running instance limit. For new users during the public beta, this limit is 20.

   If you need additional capacity, please contact us at aws@amazon.com.

10. How many instances can I launch?

    Each user has a concurrent running instance limit. For new users during the public beta, this limit is 20.

11. Can I use a static IP in my instances?

    Not at present. Your image must be configured as a DHCP client and it will be assigned an IP. Currently, all instances come with internet addressable IP addresses. If you enable access through the firewall from the "world", you can address them from anywhere.

12. How do I host a public domain if I have to DHCP an IP address?

    You can use a dynamic DNS service, such as DynDNS or ZoneEdit.

13. How do I handle time synchronization between instances?

    You can set up NTP (the Network Time Protocol) which does this for you. You can find more information at http://-www.ntp.org/. This is particularly important if you plan on using any of Amazon's web services (such as Amazon S3 or Amazon EC2) from within an instance, since requests to these services need to be timestamped.

14. Can I use my own kernel?

    Not at present.

15. Can I get a bigger/smaller/differently optimized virtual machine?

Not at present. For now, if you need more capacity launch more instances.

16. Is there a REST interface to Amazon EC2?

    Not at present. For now, you will have to use the SOAP or Query API, or the provided API command line tools.

17. How does Amazon EC2 handle load balancing?

    With a service as flexible as Amazon EC2, customers can launch any number of load balancing systems within Amazon EC2. The load balancing instances can forward traffic to other systems. There are several open source solutions that are in wide use.

18. How do I monitor my systems?

    Amazon EC2 currently only provides the most basic monitoring. You can tell from DescribeInstances whether we believe your instance is running or not. However, you may regard your systems running in Amazon EC2 as your data center, and so any monitoring instrumentation that you wish to include on the systems – be it SNMP or some other mechanism – is entirely up to you.

19. Is there any way for an instance to discover its own instance ID?

    From within your instance you can use REST-like queries to http://169.254.169.254/1.0/ to retrieve various instance specific meta-data, including the instance ID. Refer to the Developer's Guide (section 'Using Instance Data') for the details.

20. Can I pass arbitrary configuration values to an instance at launch time?

    Yes, although the size of the data is limited to 16K at the moment. Refer to the Developer's Guide for the details: section 'Using Instance Data' tells you how to retrieve data and the sections on the command-line tools and APIs tell you how to supply the data when launching an instance.

21. Why can't I retrieve my instance-specific data from within a running instance when querying http://169.254.169.254/1.0/?

    The Parameterized Launches feature is only available to instances that were launched after the feature was released. Therefore if you launched your instance before then, this data will not be available. We suggest you relaunch your instances if you want to use this functionality.

    If after relaunching your instance you still experience problems retrieving the data, you should check:

    - Are you using the correct base URI (http://169.254.169.254/1.0/)
    - Are you using the correct URI for the data you're trying to retrieve? Remember that trailing '/' may be required, depending on the data you're trying to retrieve.
    - Did you specify any launch data when launching your instances? If not you will get a HTTP error response (404) when trying to retrieve the user data. *Note* that the instance's meta-data is always available, even if you do not specify data at instance launch.

22. Is there a way to run a script on instance termination?

    Not with any reliability. Amazon EC2 tries to shut an instance down cleanly (in which case

normal system shutdown scripts will run), but there is only a short time available for things to happen and in some cases (hardware failure, for example) this does not happen. Since there is no entirely reliable way to ensure shutdown scripts run, it is best to have a strategy in place to deal with abnormal terminations.

23. Why do I get keep getting `"Request has expired"` errors?

    To reduce the risk of replay attacks our requests include a timestamp. This, along with the most important parts of the request, is signed to ensure the message (including the timestamp) can't be modified without detection.

    If the difference between the timestamp in the request and the time on our servers is larger than 5 minutes the request is deemed too old (or too new) and an error is returned.

    You need to ensure that your system clock is accurate and configured to use the correct timezone. NTP is a good way to do this.

24. How can I allow other people to launch my AMIs?

    You can allow other users to launch your AMIs by modifying the AMI's launchPermission attribute. It is possible to either grant public launch permissions, which gives all users permission to launch the AMI, or to only grant launch permissions to specific users.

    To grant public launch permissions:

    ```
    PROMPT> ec2matt ami-5bae4b32 -t launchPermission -a -i group=all
    ```

    To grant a specific user launch permissions:

    ```
    PROMPT> ec2matt ami-5bae4b32 -t launchPermission -a -i
    userId=495219933132
    ```

    To clear additional launch permissions for an AMI:

    ```
    PROMPT> ec2ratt ami-5bae4b32 -t launchPermission
    ```

25. Can I charge other people for using my AMI?

    Not at present.

26. Why do I need to reregister a rebundled AMI? Can't I keep the same AMI ID?

    An AMI ID is associated with the physical bits in an image. To protect users from images being modified we require you to reregister AMIs when rebundling.

27. Can I pass JVM properties to the command line tools?

    Yes. By setting the environment variable `EC2_JVM_ARGS` arbitrary JVM properties can be passed to the command line tools.

28. Can I use a proxy with the command line tools?

    Yes. By passing in JVM properties via the `EC2_JVM_ARGS` environment variable, proxy settings can be specified for the command line tools. For example in Linux:

```
export EC2_JVM_ARGS="-Dhttp.proxyHost=http://my.proxy.com -Dht-
tp.proxyPort=8080"
```

The following properties are supported for configuring a proxy:

Setting	Description
https.proxyHost	HTTPS proxy host
https.proxyPort	HTTPS proxy port
http.proxyHost	HTTPS proxy host
http.proxyPort	HTTPS proxy port
http.proxyRealm	Proxy realm (https and http)
http.proxyUser	Proxy username (https and http)
http.proxyPass	Proxy password (https and http)

Note

`https.proxyHost` should be used when `EC2_URL` points to an https host, and `http.proxyHost` when `EC2_URL` points to an http host.

Glossary

Glossary

Amazon Machine Image (AMI)	An Amazon Machine Image (AMI) is an encrypted machine image stored in Amazon S3. It contains all the information necessary to boot instances of your software.
Explicit Launch Permission	Launch permission granted to a specific user.
Instance	Once an AMI has been launched, the resulting running system is referred to as an instance. All instances based on the same AMI start out identical and any information on them is lost when the instances are terminated or fail.
Group	A set of customer instances that have been designated by the customer as being related by assigning them the same security group when the instances were first run. The Amazon EC2 firewall controls access to instances based on the instance's group membership and the rules defined for the group.
Launch Permission	AMI attribute allowing users to launch an AMI
Public AMI	An AMI that all users have launch permissions for.
Reservation	A collection of instances started as part of the same launch request.
Shared AMI	An AMI that users other than the owner have launch permissions for.