Authenticated Access for OTOY AMIs

For first time users, we highly recommend starting with the Octane Cloud Workstation Autodesk Edition AMI. Unlike the other, more developer focused AMIs, this AMI is meant to be more user friendly. It features a user mode desktop with full administrator privileges to install new software ‘out of the box’.

The ORBX cloud console AMIs, by contrast, are for more advanced developers who are ready to port their application or game to an isolated sandbox environment. The ORBX Cloud Console, absent developer intervention, will simply load a locked down desktop in 1024x768 with a black background.

The Octane Cloud Workstation AMI will boot directly into a sandboxed user mode Windows or Ubuntu Linux desktop with limited privileges. It is up to the administrator to expand default user privileges; the ADSK AMI is an exception, as it requires administrator rights in the user mode desktop to run the built in Autodesk applications and allow users to install plug-ins and 3rd party software.

We strongly recommend that first time users log in and familiarize themselves with the OTOY/AWS support forums on otoy.com.

Connecting to an OTOY Amazon Machine Image

Connections to OTOY AMIs are protected by a security token, which is validated before allowing a connection to complete. There are two methods for supplying the security token available.

User Supplied Token

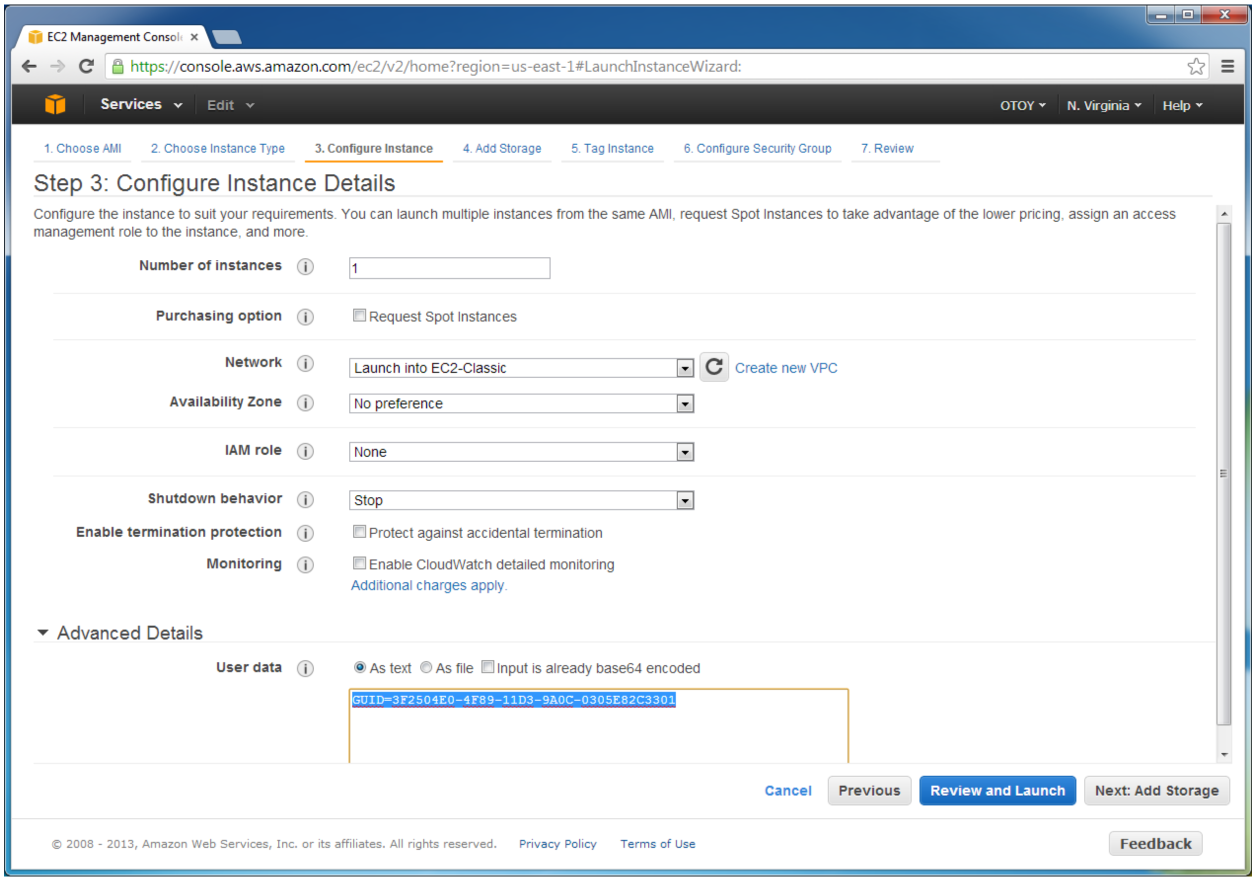
The user may supply a token when the instance is started, by including the following text in the instance metadata userdata field:

```
GUID=<guid>
```

Where the <guid> field is replaced by a randomly generated GUID token. If you require other data to be present in your userdata field, then the GUID field should come first, and must be separated from any other data by an ampersand (&) character, a semicolon (;) character or a newline character.

> *Note that case is important, and any letters should be the same case as they are supplied to the host.*
> *Note also that if a user supplied token is present, then it overrides the automatic generated token.*

The following screenshot shows an example of launching an instance through the Amazon web console, and supplying the userdata.



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1
Purchasing option	☐ Request Spot Instances
Network	Launch into EC2-Classic (Create new VPC)
Availability Zone	No preference
IAM role	None
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.

Advanced Details

User data	[✓ As text] As file ☐ Input is already base64 encoded

```
GUID=3F2504E0-4F89-11D3-9A0C-0305E82C3301
```

For good security, the GUID should be at least 128 bits long, and generated with a well seeded cryptographically secure random number generator.

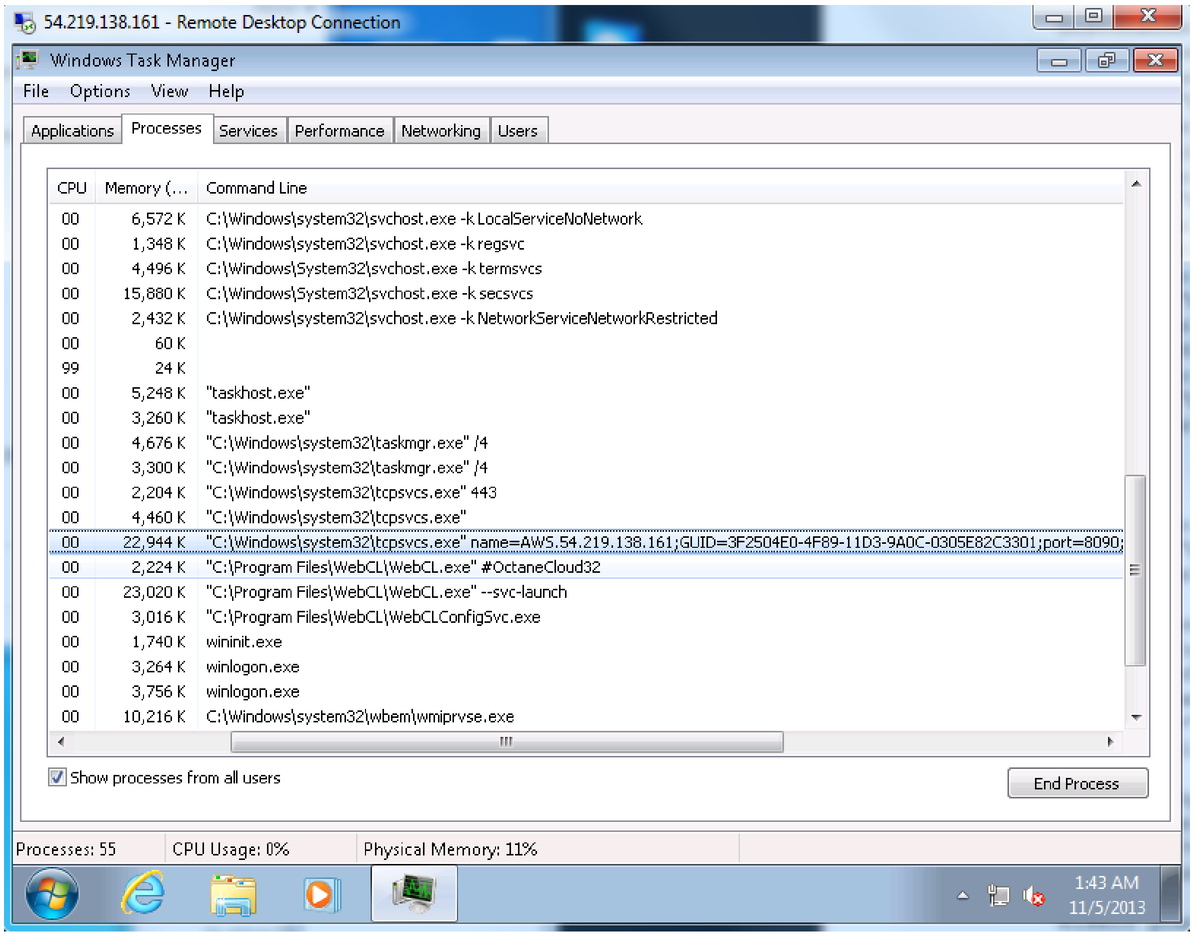
PLEASE NOTE - If you supply your own GUID in the metadata tag, the GUID generated in the command line (see below) is completely bypassed. In this case, you should ONLY be using the metadata GUID to connect, not the one in the command line. You will also need to manually supply the security group policy as well to make sure the following TCP ports are open:

```
21
80 (HTTP)
443 (HTTPS)
```

```
1000 - 1100
3389 (RDP)
5090
5900 - 5910
8090 - 8300
```

Automatic Token Generation

By default, the system will generate a secure token and supply it to the instance when the instance is launched. You must then RDP into the instance, and you can find the token by looking at the process command line in task manager, as illustrated in the following screenshot. **If a User Supplied Token exists, then the automatically generated token will not be used, but it will still show in the process command line.**



54.219.138.161 - Remote Desktop Connection
Windows Task Manager
File Options View Help
Applications Processes Services Performance Networking Users
Show processes from all users
End Process
1:43 AM
11/5/2013

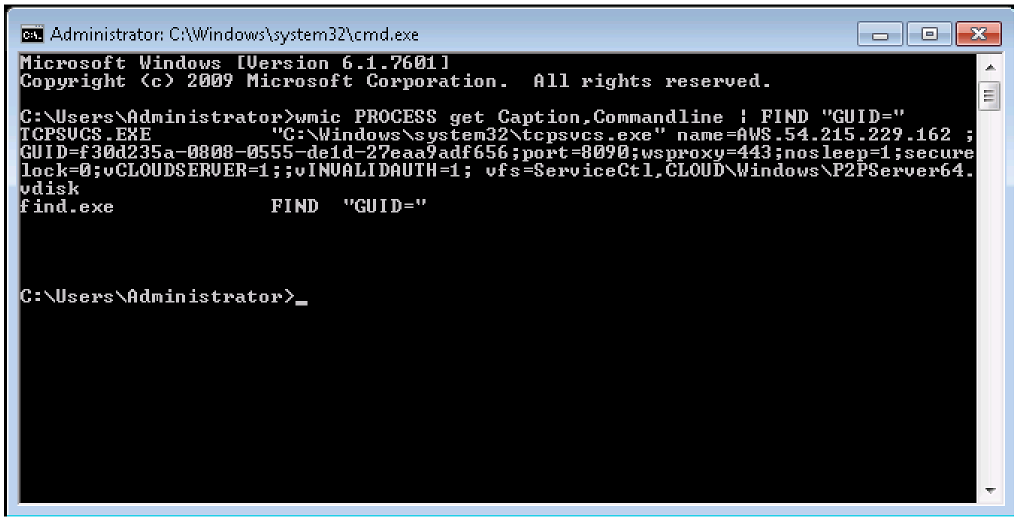
CPU	Memory (...	Command Line
00	6,572 K	C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork
00	1,348 K	C:\Windows\system32\svchost.exe -k regsvc
00	4,496 K	C:\Windows\System32\svchost.exe -k termsvcs
00	15,880 K	C:\Windows\System32\svchost.exe -k secsvcs
00	2,432 K	C:\Windows\system32\svchost.exe -k NetworkServiceNetworkRestricted
00	60 K	
99	24 K	
00	5,248 K	"taskhost.exe"
00	3,260 K	"taskhost.exe"
00	4,676 K	"C:\Windows\system32\taskmgr.exe" /4
00	3,300 K	"C:\Windows\system32\taskmgr.exe" /4
00	2,204 K	"C:\Windows\system32\tcpsvcs.exe" 443
00	4,460 K	"C:\Windows\system32\tcpsvcs.exe"
00	22,944 K	"C:\Windows\system32\tcpsvcs.exe" name=AWS.54.219.138.161;GUID=3F2504E0-4F89-11D3-9A0C-0305E82C3301;port=8090;
00	2,224 K	"C:\Program Files\WebCL\WebCL.exe" #OctaneCloud32
00	23,020 K	"C:\Program Files\WebCL\WebCL.exe" --svc-launch
00	3,016 K	"C:\Program Files\WebCL\WebCLConfigSvc.exe
00	1,740 K	wininit.exe
00	3,264 K	winlogon.exe
00	3,756 K	winlogon.exe
00	10,216 K	C:\Windows\system32\wbem\wmiprvse.exe

Processes: 55 CPU Usage: 0% Physical Memory: 11%

You can also find the token from the command shell, by executing the following from a cmd.exe shell prompt:

```
wmic PROCESS get Caption,Commandline | FIND "GUID="
```

Which can be seen in the following screenshot:



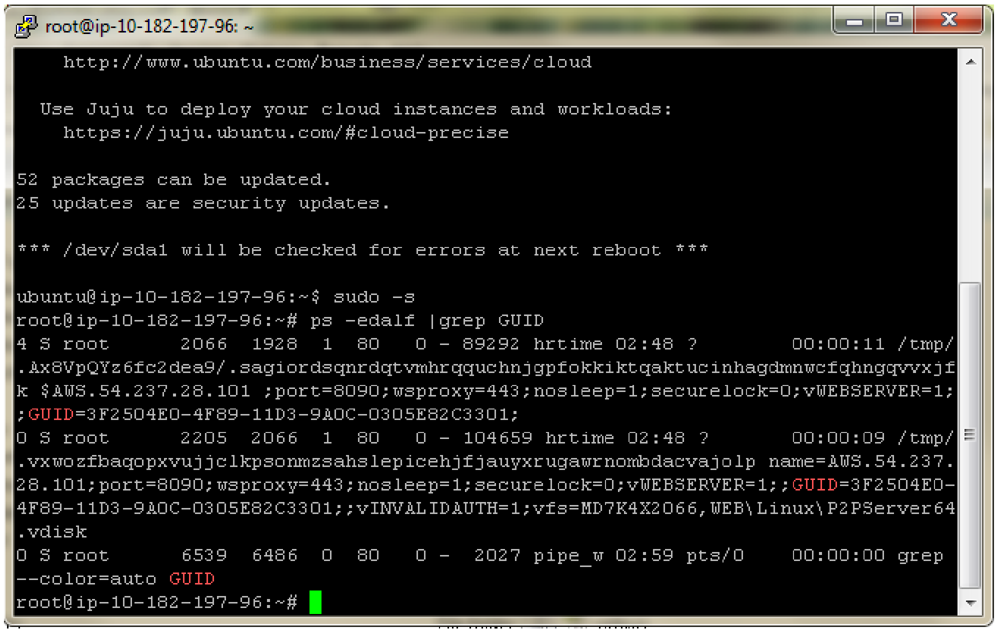
Administrator: C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Administrator>wmic PROCESS get Caption,Commandline | FIND "GUID="
TCPSVCS.EXE          "C:\Windows\system32\tcpsvcs.exe" name=AWS.54.215.229.162 ;
GUID=f30d235a-0808-0555-de1d-27eaa9adf656;port=8090;wsproxy=443;nosleep=1;secure
lock=0;vCLOUDSERVER=1;;vINVALIDAUTH=1; vfs=ServiceCtl,CLOUD\Windows\P2PServer64.
vdisk
find.exe             FIND  "GUID="


C:\Users\Administrator>_
```

On Linux, you can find the GUID from the console with the following command:

```
ps -edalf | grep GUID
```

Which is illustrated by the following screenshot:



root@ip-10-182-197-96: ~

```
    http://www.ubuntu.com/business/services/cloud

  Use Juju to deploy your cloud instances and workloads:
    https://juju.ubuntu.com/#cloud-precise

52 packages can be updated.
25 updates are security updates.

*** /dev/sda1 will be checked for errors at next reboot ***

ubuntu@ip-10-182-197-96:~$ sudo -s
root@ip-10-182-197-96:~# ps -edalf |grep GUID
4 S root      2066  1928  1  80   0 - 89292 hrtime 02:48 ?        00:00:11 /tmp/
.Ax8VpQYz6fc2dea9/.sagiordsqnrdqtvmhrqquchnjgpfokkiktqaktucinhagdmnwcfqhngqvvxjf
k $AWS.54.237.28.101 ;port=8090;wsproxy=443;nosleep=1;securelock=0;vWEBSERVER=1;
;GUID=3F2504E0-4F89-11D3-9A0C-0305E82C3301;
0 S root      2205  2066  1  80   0 - 104659 hrtime 02:48 ?       00:00:09 /tmp/
.vxwozfbaqopxvujjclkpsonmzsahslepicehjfjauyxrugawrnombdacvajolp name=AWS.54.237.
28.101;port=8090;wsproxy=443;nosleep=1;securelock=0;vWEBSERVER=1;;GUID=3F2504E0-
4F89-11D3-9A0C-0305E82C3301;;vINVALIDAUTH=1;vfs=MD7K4X2066,WEB\Linux\P2PServer64
.vdisk
0 S root      6539  6486  0  80   0 -  2027 pipe_w 02:59 pts/0    00:00:00 grep
--color=auto GUID
root@ip-10-182-197-96:~# 
```

Launching the Client

The client is launched from the following URL in a web browser:

```
http://aws.otoy.com/?connect=//<ip>:8090/?NOCRYPTO=1;STATUS=1;
AUTOLOCK=0;WAN=12;GUID=<guid>
```

Where the <ip> is replaced with the public IP address of the instance, and `<guid>` is replaced with the GUID token described in the previous sections.

ORBX.js minimum browser requirements

ORBX.js is supported in the following browsers on Linux, OSX and Windows, with WebGL enabled:

- Firefox 19+
- Chrome 26+
- Opera 16+
- Safari 7+

A typical quad core PC should be able to decode 720p60 streams from within Firefox or Chrome. Safari 7 can handle 720p30 streams. A good rule of thumb to measure expected ORBX performance is to assume that each core on the client PC can decode 720p30 in JavaScript.

Audio is supported in all of the above browsers, but must be turned on by adding AUDIO=1; to the connection string.

The default bandwidth cap is set to 12 Mbps, which is enough to cover a 1080p stream at good quality for normal desktop applications. You can change this setting when you connect to the host by setting WAN=n, where n is the bandwidth cap in Mbps. The minimum bandwidth for a 60 fps ORBX stream is 3 Mbps. If you are on a 2 Mbps line for example, you should use “WAN=2;MAXFPS=30;” to cap the stream at 2 Mbps and 30 fps.

When using native ORBX, with UDP, the bitrate does not need to be set, as the ORBX stream automatically detects the instantaneous available bandwidth, and adjusts the stream on the fly. This functionality will be added to ORBX.js once browsers properly expose UDP web sockets to JavaScript.

ORBX.js offers partial support on several non-WebGL browsers - including iOS browsers and IE10/11 - through a 30 fps simplified i-frame only stream (typically 2x the bandwidth of a normal ORBX stream).

ORBX.js roadmap

ORBX.js will be refactored to support mobile browsers in early 2014, as WebGL support matures across a broad range of mobile platforms.

When WebGL 2 rolls out to the major browsers, ORBX.js will shift nearly all decoding to the GPU, and support 4k and even 8K decoding in the browser.

ORBX 2D (currently in beta) is the next major milestone for ORBX.js, with bitrates about half of H.264 high profile, and support for 12-bits of color per channel, unlimited bit planes (including depth and alpha), and vector based compositor.

Frequently Asked Questions

Why does the GUID that I entered into the metadata not show up in the host process command line?

The metadata GUID overrides the command line guid, but will not be displayed in the command line.

Why do I get an "Authentication Failure" message when I connect?

Your client GUID does not match the GUID supplied in the metadata.
Otherwise, if no GUID is supplied in the metadata, then your client GUID does not match the host process command line GUID.

What is the expected performance?

Up to 60 frames per second depending on bandwidth and CPU configuration.

Is ORBX.js client side code or server side code?

Both. The JavaScript code is run in the browser, but is loaded from a custom web server built into the OTOY Amazon Machine.

Further reading

- ORBX and ORBX.js developer forums on otoy.com: http://render.otoy.com/forum/viewforum.php?f=64
- ORBX SDK client documentation: http://aws.otoy.com/docs/OTOYClientAPI.pdf
- ORBX.js <iframe> documentation (WIP): http://aws.otoy.com/docs/ORBX.js_Iframe_API.pdf
- Octane Cloud Workstation Technical Specs: http://aws.otoy.com/docs/OctaneCloud6.pdf
- ORBX 2D Whitepaper: http://aws.otoy.com/docs/ORBX2_Whitepaper.pdf
- ORBX Cloud Gaming Roadmap: http://www.otoy.com/cgc/cloudgaming_2013.pdf
- OTOY Amazon Machine Images in the AWS Marketplace: :https://aws.amazon.com/marketplace/seller-profile/ref=dtl_pcp_sold_by?ie=UTF8&id=795808b7-f99d-426d-bb03-8aa79ff5b65e