Cloud.com CloudStack Adapter

Framework Guide

(Version 2.0 Preview)

May 5, 2010

© 2010 Cloud.com Inc. All rights reserved. Specifications are subject to change without notice. The Cloud.com logo, Cloud.com, Hypervisor Attached Storage, HAS, Hypervisor Aware Network, HAN, and VMSync are trademarks or registered trademarks of Cloud.com, Inc. All other brands or products are trademarks or registered trademarks of their respective holders.

Contents

[1 Overview 4](#_Toc235887332)

[1.1 Getting Started 4](#_Toc235887333)

[2 UserAuthenticators 5](#_Toc235887334)

# Overview

The CloudStack Adapter Framework allows you to further customize your CloudStack Software to suit your business needs. We’d like to think that our default adapter implementations should be sufficient enough to suit your needs but we also know that every business has unique requirements. This framework will allow you to build your own adapters and easily integrate within the Cloud.com CloudStack. We currently support custom implementation of the following:

* UserAuthenticators – allows you to integrate custom authentication schemes that the Management Server can use to validate user credentials.
* HostAllocators – allows you to create custom rules to determine which physical host to allocate the guest virtual machines on.
* PodAllocators – allows you to create custom rules to determine which pod to allow the guest virtual machines on.
* ConsoleProxyAllocators – allows you to create custom rules to determine which routing nodes to allow the console proxy on.
* Investigators – allows you to create custom rules to determine when a virtual machine is detected as “down” for HA purposes.

## Getting Started

The CloudStack Adapter Framework is built entirely using Java as the programming language. To get started:

* Make you have the CloudStack Management Server installed and running (If not, please refer to the Cloud.com CloudStack Installation Guide)
* Understanding of XML (Refer to the W3C XML Tutorial)
* Understanding of the Java programming language.
* Understanding of how the Cloud.com CloudStack works.

# UserAuthenticators

User authenticators are written by extending the com.cloud.server.auth.DefaultUserAuthenticator interface. The interface contains one method.

/\*\*

\* Authenticates the user by username and password.

\*

\* **@param** username

\* **@param** password

\* **@param** domainId

\* **@return** true if the user has been successfully authenticated, false otherwise

\*\*/

**public** **boolean** authenticate(String username, String password, Long domainId);

A custom adapter can be written by implementing this one method. Here’s an example custom authenticator:

import javax.ejb.Local;

@Local(value={com.cloud.server.auth.UserAuthenticator.class})

public class MyAuthenticator extends com.cloud.server.auth.DefaultUserAuthenticator {

public static final Logger s\_logger = Logger.getLogger(MyAuthenticator.class);

**public** **boolean** authenticate(String username, String password, Long domainId) {

**if** (*s\_logger*.isDebugEnabled()) {

*s\_logger*.debug("Retrieving user: " + username);

}

UserAccount user = \_userAccountDao.getUserAccount(username, domainId);

**if** (user == **null**) {

*s\_logger*.debug("Unable to find user with " + username + " in domain " + domainId);

**return** **false**;

}

MessageDigest md5;

**try** {

md5 = MessageDigest.*getInstance*("MD5");

} **catch** (NoSuchAlgorithmException e) {

**throw** **new** CloudRuntimeException("Error", e);

}

md5.reset();

BigInteger pwInt = **new** BigInteger(1, md5.digest(password.getBytes()));

// make sure our MD5 hash value is 32 digits long...

StringBuffer sb = **new** StringBuffer();

String pwStr = pwInt.toString(16);

**int** padding = 32 - pwStr.length();

**for** (**int** i = 0; i < padding; i++) {

sb.append('0');

}

sb.append(pwStr);

**if** (!user.getPassword().equals(sb.toString())) {

*s\_logger*.debug("Password does not match");

**return** **false**;

}

**return** **true**;

}

}

Once this class has been written and your code has been packaged into a JAR file, copy the jar file to /usr/share/java. Then modify components.xml & components-premium.xml found in /usr/share/cloud/management/conf. Look for the following in both files.  
  
<adapters key="com.cloud.server.auth.UserAuthenticator">  
 <adapter name="MD5" class="com.cloud.server.auth.MD5UserAuthenticator"/>  
</adapters>

Replace the MD5UserAuthenticator with your class name (you can change the name of the adapter as well).   
You will also need to modify /usr/share/cloud/management/etc/tomcat6.conf and append the path and name of your jar to the end of the CLASSPATH.

Once this has been completed, restart the Management Server by typing “service cloud-management restart”.