Cisco CloudCenter Integration Guide, Release 4.5.x

First Published: July 15, 2016

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: http://www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

Integrations

ACI Integration

CloudCenter users can create simple, out-of-the-box application profiles where they can create infrastructure-independent models of any application. Once modeled, CliQr and Cisco work together to provide automated, end-to-end provisioning of compute, storage, and Application Centric Infrastructure (ACI) network configurations as well as the application and its set of required components – including middleware and application data or packages based on the application's specific needs, characteristics and dependencies – all in one click and onto any physical, virtual, or cloud environment.

A CliQr CloudCenter integration with Cisco ACI and VMware vSphere requires the following minimum resources to be configured for each area.

Component	Requirement	Details
Cisco ACI	CliQr automates the end-to-end-provisioning of the overlay infrastructure and deployments of applications.	On ACI this includes the provisioning and management of the following resources: • Application Network Profiles (ANP) • Endpoint Groups (EPG) • Contracts • Subjects/Filters
	A working Cisco ACI environment with the following applications configured.	As a prerequisite for CliQr CloudCenter to provision and configure the applications on APIC, the initial setup must be completed. • Leaf switch profiles, Switch Selectors, Interface Profile, and Policy Groups • VLAN Pool • VMware's Virtual Machine Manager (VMM) Domain • Routable IP subnet to a New Tenant and Bridge Domain(s) configured with L3 Out for external internet connectivity • Routing protocols • VRF • Verify that the setup has passed all checks to ensure end-to-end connectivity.
	Valid APIC SSL certificate must be associated to the host name. Or HTTP access must be enabled.	By default, the APIC controller only listens to HTTPS for both the GUI and REST APIs. For ACI integration to work via APIs, ensure these tasks. • The APIC SSL certificate is valid for the APIC host name. • No valid certificate exists in the APIC controller. • The APIC controller is accessible using an IP address. • HTTP access is enabled to the APIC controller.
	Using the APIC GUI, manually add a new application network profile with one EPG.	CliQr requires ***access credentials*** to the APIC controller, tenant name, and VMM name.
	Verify that the APIC controller is provisioned with a new Virtual Distributed Switch (vDS) port group.	

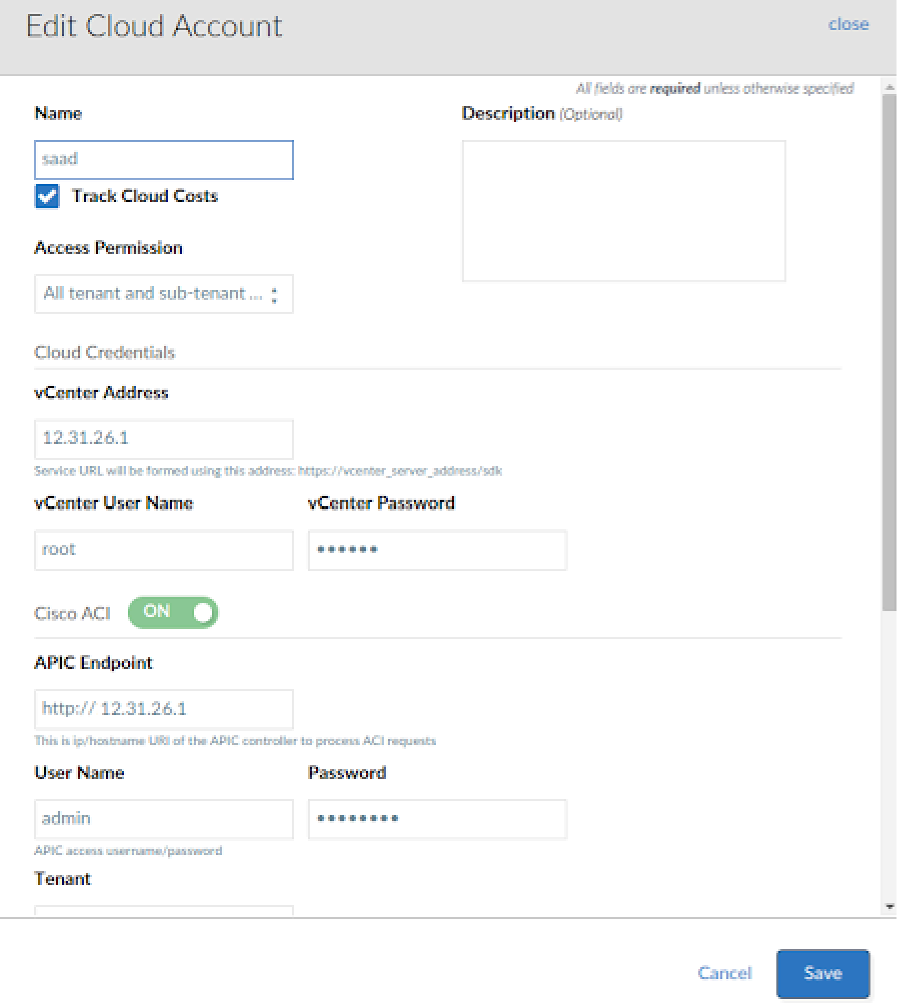
	Using the vCenter UI, provision/clone a new VM with the network pointing to the created portgroup.	
	SSH/Console into new VM and verify that outbound internet access works.	For example: **curl -L http://google.com** (with or without a proxy)
VMware vSphere	A working VMware vCenter 5.0/5.5 environment	The minimum VMware vSphere version is v5.0, but CliQr recommends vSphere v5.5 U2.
	CliQr automates the provisioning of virtual machines into the VMware private datacenter.	CliQr requires ***access credentials*** to the vCenter setup.
	All ESX host(s) must be physically connected to the ACI leaf switches.	As a prerequisite for the CliQr installation, the minimum requirements for the datacenter are: • A physical ESX host capable of running at least 10 medium sized instances • An ESX cluster (cluster could comprise of just the one host) • A datastore (or datastore cluster for DRS support), at least 100gb of free space
	If the ESXi hosts are Cisco UCS based	• The VLANs for the CMM must be mapped to the vNIC template. • The uplinks from the Fabric must interconnect trunking VLANs to the leaf switches.
CliQr CloudCenter	CliQr automates and optimizes the provisioning of infrastructure and deployments of new or existing applications. See Networking Requirements for a list of all the ports for each component.	Dedicated Deployment Models require four virtual appliances. • The CCM Appliance: manage clouds, environments, and applications (UI / REST) • The CCO Appliance: provisions compute, storage, networking, and security for virtual or physical environment. • The AMQP Appliance (Rabbit): communication between the application VMs and the CCO. • The Application VM Base OS Images: CentOS/Ubuntu Linux (or Windows) CliQr requires: • The IP address for the CCM, CCO, and AMQP servers. • The HTTP/HTTPS proxy address if using HTTP proxy for external Internet access. • OVAs if you are using your own application VM images (CliQr needs to install the Management Agent on each VM.
	The CloudCenter Virtual Appliances are for Linux CentOS 6.5 (7.0 is supported)	See Install CloudCenter
	VMs distributed as single-file OVAs must be imported into the VMware datacenter.	CliQr requires ***access credentials*** to the vCenter setup.
	CliQr provides both a CentOS 6.5 and Ubuntu 12.04 Base OS Images for application VMs, but customers may use their own CentOS/RHEL/Ubuntu image, or for Windows workloads, a Windows 2K8/2K12 image.	Dynamically Bootstrap Custom Images
	CliQr CloudCenter also provides IP address management (IPAM).	Application VMs could receive IP addresses one of the three ways. • InfoBlox (CliQr requires ***access credentials*** to the InfoBlox setup.) • DHCP server • Customizable script based IP assignment (need IP address, subnet, and ranges)

	The application VMs require external Internet access to download middleware services.	For example: apt-get install mysql/yum install mysql

When you have configured the resources and tasks listed in the section above, including installing CloudCenter, verifying network connectivity all around, and launching application to ensure everything is working from end-to-end, then you are ready to configure the ACI integration in the CCM UI.

To configure the ACI integration in the CCM UI, follow this procedure.

1. Access the CCM UI and click **Admin** > **Clouds**.
2. Click **Configure Cloud**.
3. Edit an existing VMware cloud or add a new VMware cloud as required for your ACI integration.



Edit Cloud Account

All fields are ***required*** *unless otherwise specified*

Name: saad

☑ Track Cloud Costs

Description *(Optional)*:

Access Permission: All tenant and sub-tenant ...

Cloud Credentials

vCenter Address: 12.31.26.1

Service URL will be formed using this address: https://vcenter_server_address/sdk

vCenter User Name: root

vCenter Password: ••••••

Cisco ACI: ON

APIC Endpoint: http:// 12.31.26.1

This is ip/hostname URI of the APIC controller to process ACI requests

User Name: admin

Password: ••••••••

APIC access username/password

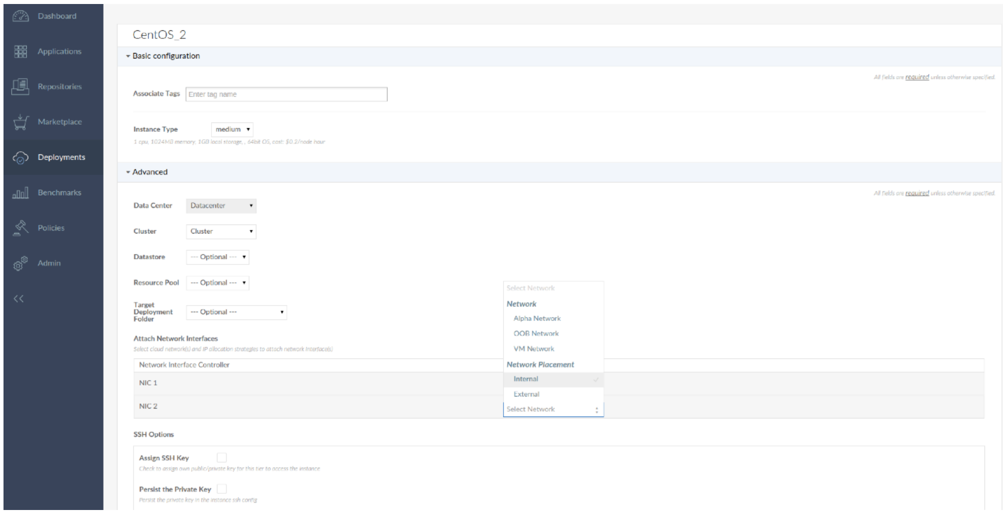
Tenant

For the APIC controller's endpoint, configure the following information:
- The host name or the IP address of where the APIC controller resides in the Endpoint.
- The access credentials for the endpoint.
- The CloudCenter tenant for this ACI.
- The VMM domain configured in the Cisco ACI setup.

4. Click **Save**. The ACI cloud configuration information is now saved.

To launch the ACI integration in your cloud, follow this procedure.

1. Click the application that is configured with the ACI integration and select **Edit** from the dropdown list.
2. Scroll down to the Advanced section where you list the NIC cards.



CentOS_2

Basic configuration

All fields are required unless otherwise specified.

Associate Tags: Enter tag name

Instance Type: medium

1 cpu, 1024MB memory, 1GB local storage, , 64bit OS, cost: $0.2/node hour

Advanced

All fields are required unless otherwise specified.

Data Center: Datacenter

Cluster: Cluster

Datastore: --- Optional ---

Resource Pool: --- Optional ---

Target Deployment Folder: --- Optional ---

Attach Network Interfaces

Select cloud network(s) and IP allocation strategies to attach network interface(s)

Network Interface Controller

NIC 1

NIC 2

SSH Options

Assign SSH Key □

Check to assign own public/private key for this tier to access the instance

Persist the Private Key □

Persist the private key in the instance ssh config

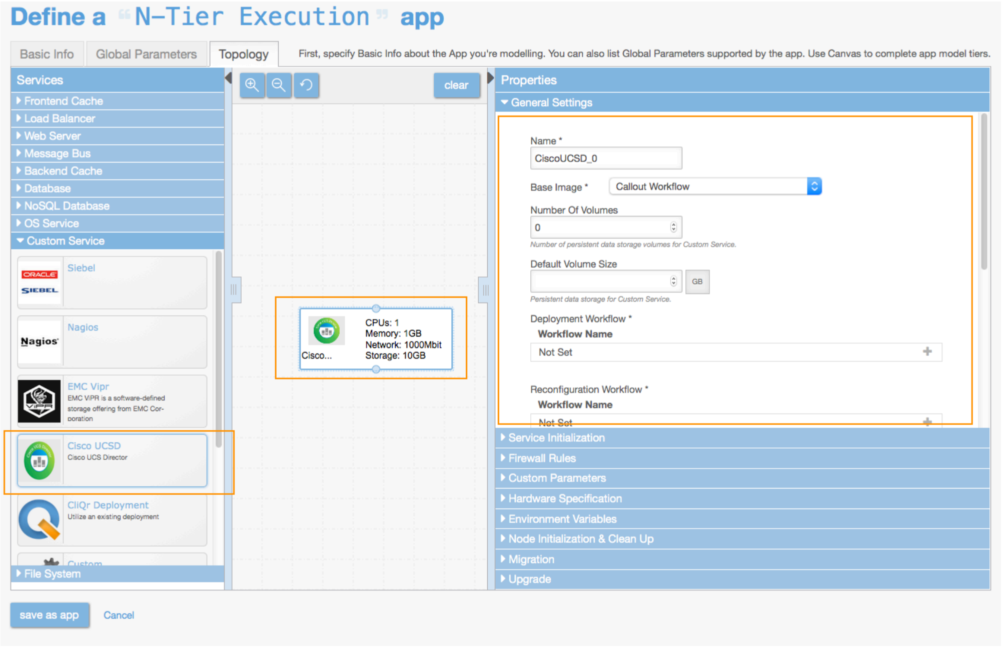
3. In the Attach Network Interface section, list the actual VMware NIC cards.
4. Designate a Bridge domain from the ACI environment. The list of bridge domains is pulled from ACI.

Cisco UCSD Integration

- Overview
- Limitations
- Integration Requirements
- UCSD Workflow Support
- Configure UCSD as a Custom Service
- Other References

CliQr provides Cisco Unified Computing Systems Director (UCSD) integration that enables you to invoke UCSD callout workflows. Users can drag and drop the Cisco UCSD service into the CloudCenter Topology Modeler and create a topology with single or multiple UCSD callouts. This allows enterprises to create a mixed topology of applications using UCSD callouts and provides the following benefits:

- Enterprises can use CloudCenter for governance as well as workflow management.
- SysAdmins can use UCSD to provision physical storage.



First, specify Basic Info about the App you're modelling. You can also list Global Parameters supported by the app. Use Canvas to complete app model tiers.

EMC ViPR is a software-defined storage offering from EMC Corporation

Cisco UCS Director

Utilize an existing deployment

CPUs: 1
Memory: 1GB
Network: 1000Mbit
Storage: 10GB

Name *: CiscoUCSD_0

Base Image *: Callout Workflow

Number Of Volumes: 0

Number of persistent data storage volumes for Custom Service.

Default Volume Size

Persistent data storage for Custom Service.

Deployment Workflow *

Workflow Name: Not Set

Reconfiguration Workflow *

Workflow Name: Not Set

Be aware of the following limitations if you decided to use this integration:

- This implementation of the USCD integration allows you to provision your storage setup on network appliances (tested and verified by CliQr).

> ⚠ UCSD currently allows VM provisioning (not tested and verified by CliQr).

- CliQr has tested and implemented this feature for select customers.
- This version only supports the integration that is explicitly mentioned in this page.
- Worker1 appliances are not required for this integration.
- CCM and CCO appliances are not available for this integration.

To integrate with Cisco UCSD, the CloudCenter SysAdmin must adhere to the following requirements:

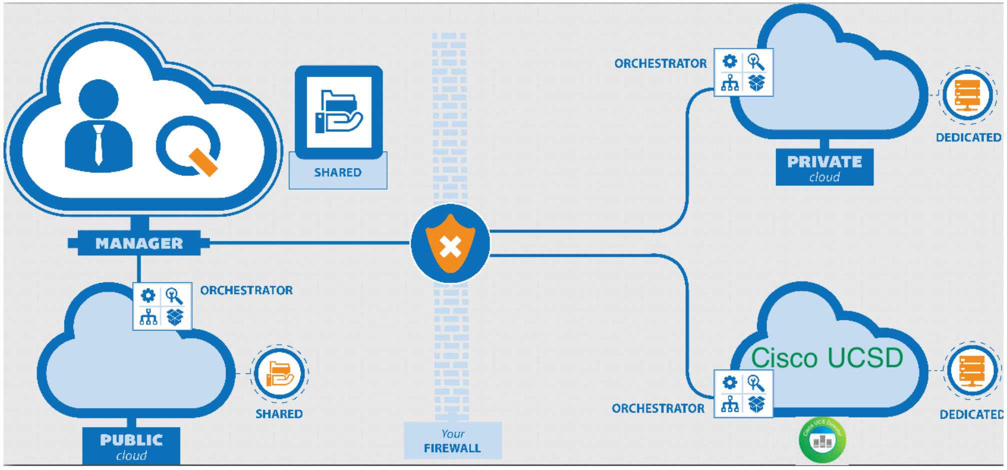
- Administrator ability to access to the Cisco UCSD account and environment.

> ⚠ If you intend to integrate UCSD in your enterprise, CliQr needs to access your UCSD environment to provide end-to-end deployment.

- Knowledge of the list of UCSD workflows to be called by the CloudCenter platform.

> ✓ CliQr abstracts the orchestration process to use the callout flows exposed by UCSD. CliQr is not aware of internal UCSD contents or tasks. The CloudCenter platform merely uses exposed UCSD parameters for cloud governance and management purposes.

- Each Cisco UCSD instance must be associated with a one CCO before you Model an Application.



- Currently, one CloudCenter platform supports one UCSD instance.
- Each CloudCenter UCSD implementation requires an associated physical image entry in the CloudCenter platform (this is a dummy placeholder – even if a logical Image is not used).

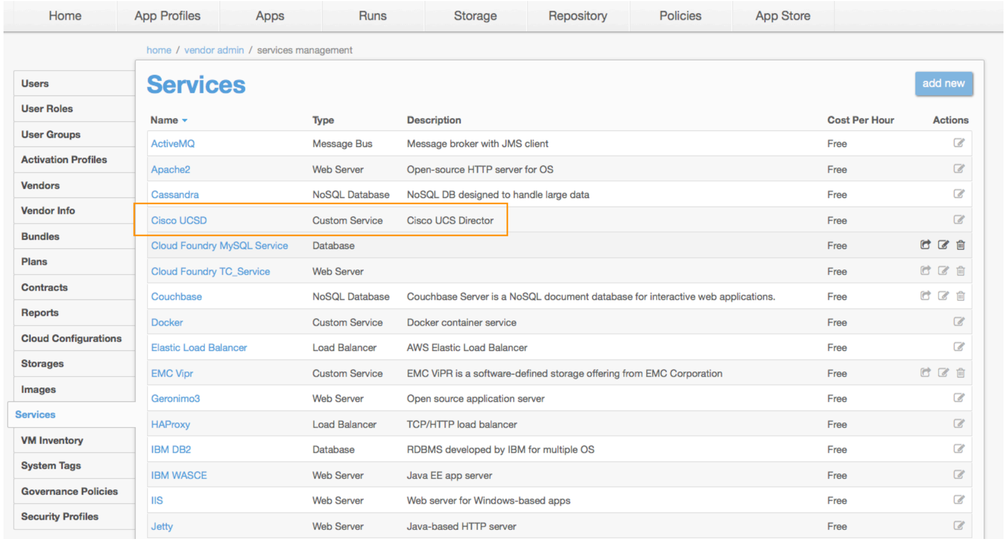
UCSD has the concept of workflows. These workflows differ between enterprises and deployments. An example of such a workflow is when you provision storage using the UCSD integration, CloudCenter currently calls the following workflows:

- Create a new storage space
- Validate the existence of a storage space
- Update an existing storage space
- Delete a storage space
- Retrieve information about a storage space

The current UCSD workflows are specific to the creation and maintenance of additional storage work spaces for enterprises. As UCSD is associated with one CCO, each time you access information about the storage space, the CCO retrieves the permitted UCSD workflows.

To configure, define, and use UCSD as a custom service, follow this process:

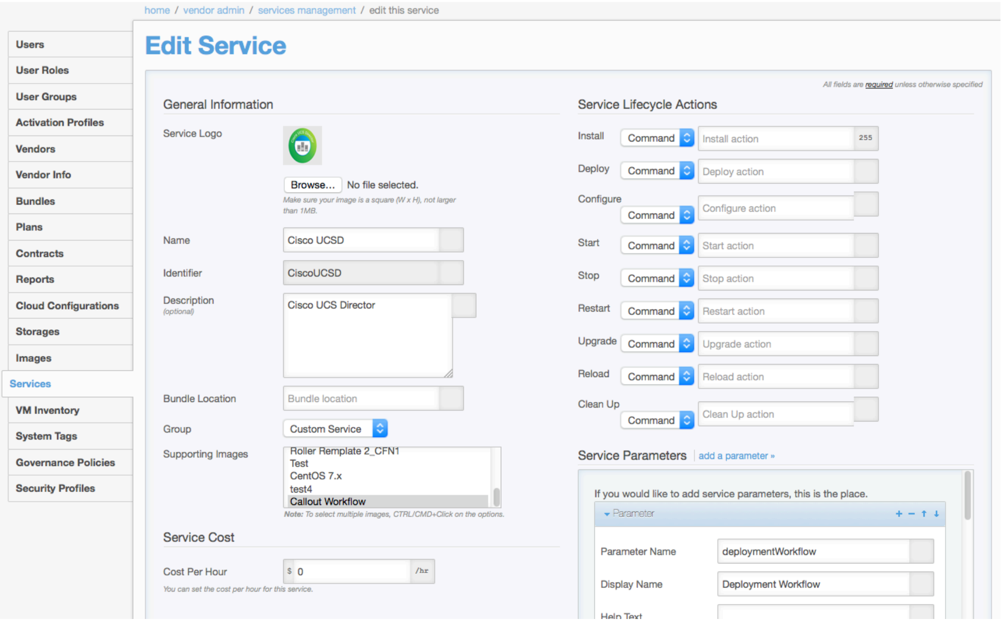
1. Configure the UCSD cloud. See Configure Cloud(s).
   Cloud Type: **UCSD**
   Cloud Family: **Cisco UCSD**
   Region: User configurable names
2. Configure a dummy UCSD physical image (see Configure the Image ID) and Map this Image.
3. Edit the **Cisco UCSD** service to define the allowed UCSD workflow parameters. See Define a Custom Service.



home / vendor admin / services management

Services

Name	Type	Description	Cost Per Hour	Actions
ActiveMQ	Message Bus	Message broker with JMS client	Free	
Apache2	Web Server	Open-source HTTP server for OS	Free	
Cassandra	NoSQL Database	NoSQL DB designed to handle large data	Free	
Cisco UCSD	Custom Service	Cisco UCS Director	Free	
Cloud Foundry MySQL Service	Database		Free	
Cloud Foundry TC_Service	Web Server		Free	
Couchbase	NoSQL Database	Couchbase Server is a NoSQL document database for interactive web applications.	Free	
Docker	Custom Service	Docker container service	Free	
Elastic Load Balancer	Load Balancer	AWS Elastic Load Balancer	Free	
EMC Vipr	Custom Service	EMC ViPR is a software-defined storage offering from EMC Corporation	Free	
Geronimo3	Web Server	Open source application server	Free	
HAProxy	Load Balancer	TCP/HTTP load balancer	Free	
IBM DB2	Database	RDBMS developed by IBM for multiple OS	Free	
IBM WASCE	Web Server	Java EE app server	Free	
IIS	Web Server	Web server for Windows-based apps	Free	
Jetty	Web Server	Java-based HTTP server	Free	



home / vendor admin / services management / edit this service

Edit Service

All fields are required unless otherwise specified

General Information

Service Logo: No file selected.

Make sure your image is a square (W x H), not larger than 1MB.

Name: Cisco UCSD

Identifier: CiscoUCSD

Description *(optional)*: Cisco UCS Director

Bundle Location: Bundle location

Group: Custom Service

Supporting Images: Roller Remplate 2_CFN1, Test, CentOS 7.x, test4, Callout Workflow

Note: To select multiple images, CTRL/CMD+Click on the options.

Service Cost

Cost Per Hour: $ 0 /hr

You can set the cost per hour for this service.

Service Lifecycle Actions

Install: Command — Install action — 255

Deploy: Command — Deploy action

Configure: Command — Configure action

Start: Command — Start action

Stop: Command — Stop action

Restart: Command — Restart action

Upgrade: Command — Upgrade action

Reload: Command — Reload action

Clean Up: Command — Clean Up action

Service Parameters

If you would like to add service parameters, this is the place.

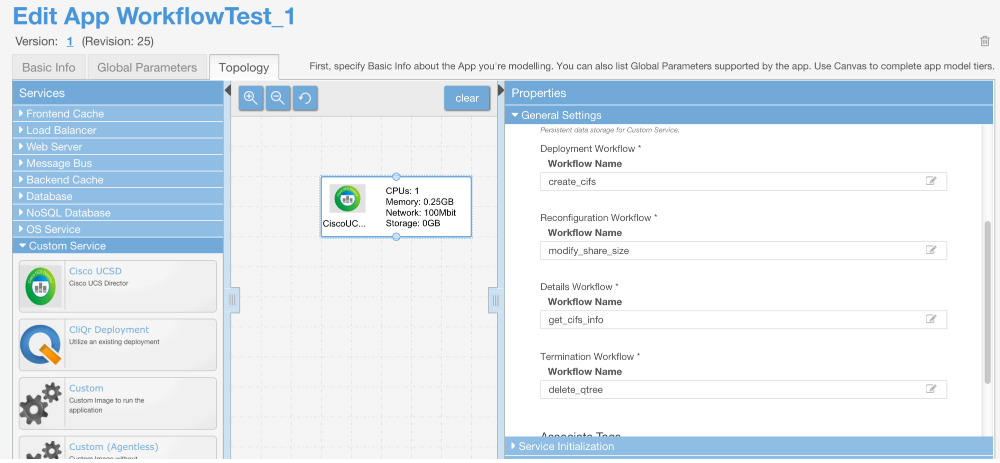
Parameter

Parameter Name: deploymentWorkflow

Display Name: Deployment Workflow

Help Text

4. Model an Application to use the UCSD service. The defined UCSD service parameters are displayed within the Custom Service section in the Topology Services page (on the right pane), when you select the UCSD service.



Edit App WorkflowTest_1

Version: 1 (Revision: 25)

First, specify Basic Info about the App you're modelling. You can also list Global Parameters supported by the app. Use Canvas to complete app model tiers.

CPUs: 1
Memory: 0.25GB
Network: 100Mbit
Storage: 0GB

Persistent data storage for Custom Service.

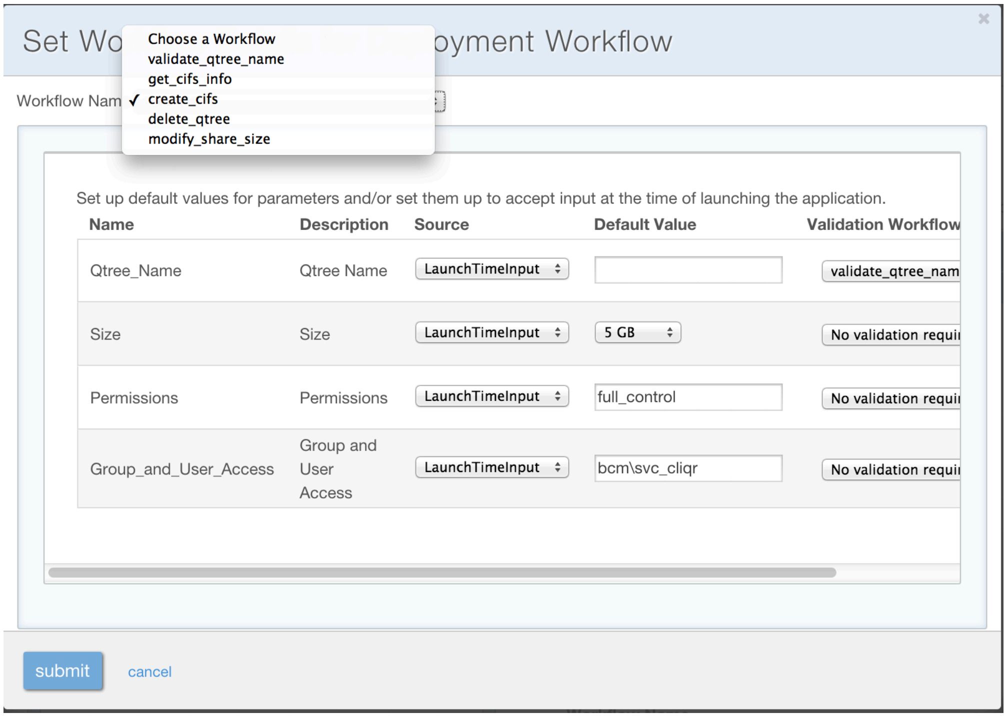
Deployment Workflow *
Workflow Name: create_cifs

Reconfiguration Workflow *
Workflow Name: modify_share_size

Details Workflow *
Workflow Name: get_cifs_info

Termination Workflow *
Workflow Name: delete_qtree

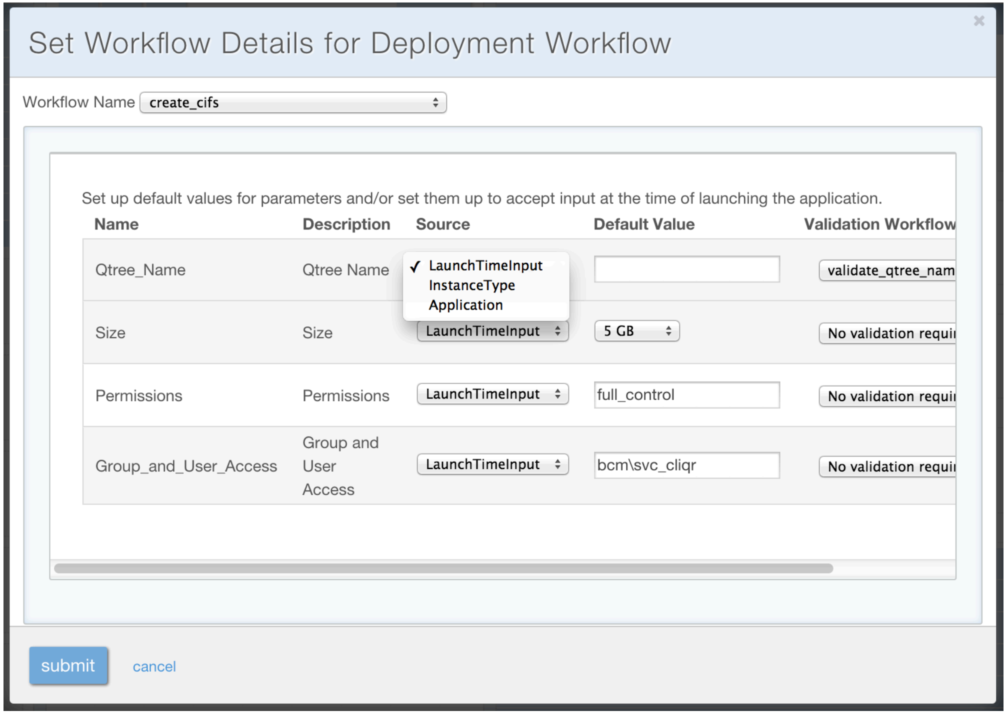
5. If required, you can add additional Global Parameters. See Topology Modeler > Global Parameters.
6. The list of configured UCSD workflows are displayed in the General Settings section. When you click a configured workflow, you see the parameters associated with that particular workflow.



Set up default values for parameters and/or set them up to accept input at the time of launching the application.

Name	Description	Source	Default Value	Validation Workflow
Qtree_Name	Qtree Name	LaunchTimeInput		validate_qtree_nam
Size	Size	LaunchTimeInput	5 GB	No validation requi
Permissions	Permissions	LaunchTimeInput	full_control	No validation requi
Group_and_User_Access	Group and User Access	LaunchTimeInput	bcm\svc_cliqr	No validation requi

7. Select the Source for each parameter in this workflow from the dropdown list:
   - **LaunchTimeinput**: Indicates that this parameter can be overridden when users deploy the application.
   - **InstanceType**: When you select the Instance Type during application deployment, the storage size associated with the Instance Type becomes the value for this parameter.
   - **Application**: Enter a default value to be used at deployment time (cannot be overridden).

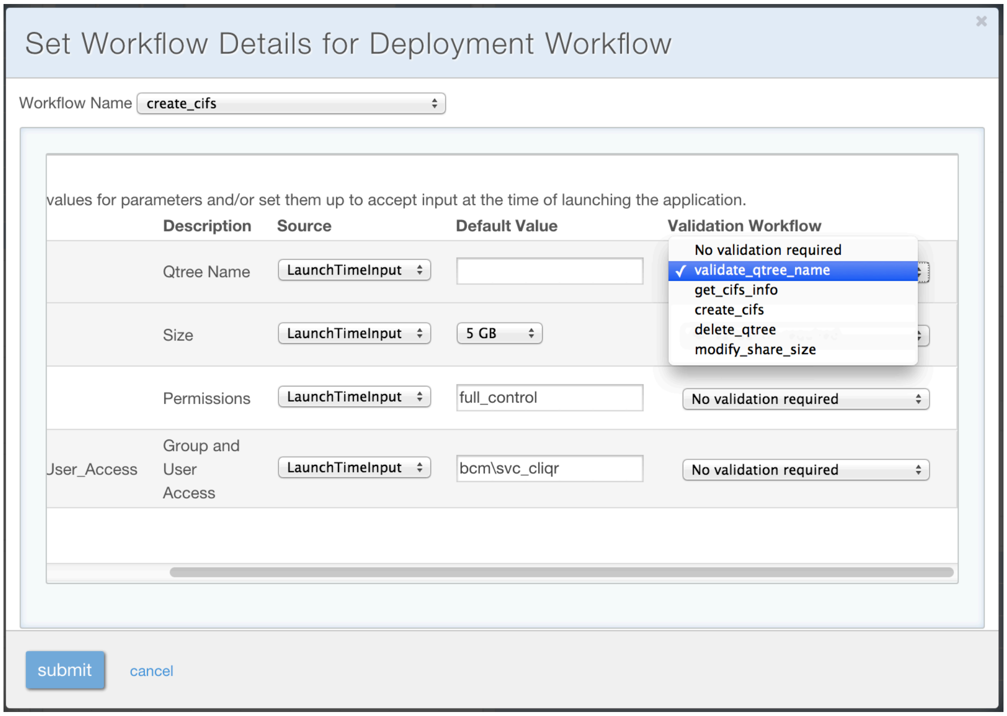


Workflow Name create_cifs

Set up default values for parameters and/or set them up to accept input at the time of launching the application.

Name	Description	Source	Default Value	Validation Workflow
Qtree_Name	Qtree Name	LaunchTimeInput		validate_qtree_nam
Size	Size	LaunchTimeInput	5 GB	No validation requi
Permissions	Permissions	LaunchTimeInput	full_control	No validation requi
Group_and_User_Access	Group and User Access	LaunchTimeInput	bcm\svc_cliqr	No validation requi

8. If applicable, you can also select the required Validation Workflow from the dropdown list for each parameter. This list is specific to each deployment and can also be filtered to only display the applicable workflows.



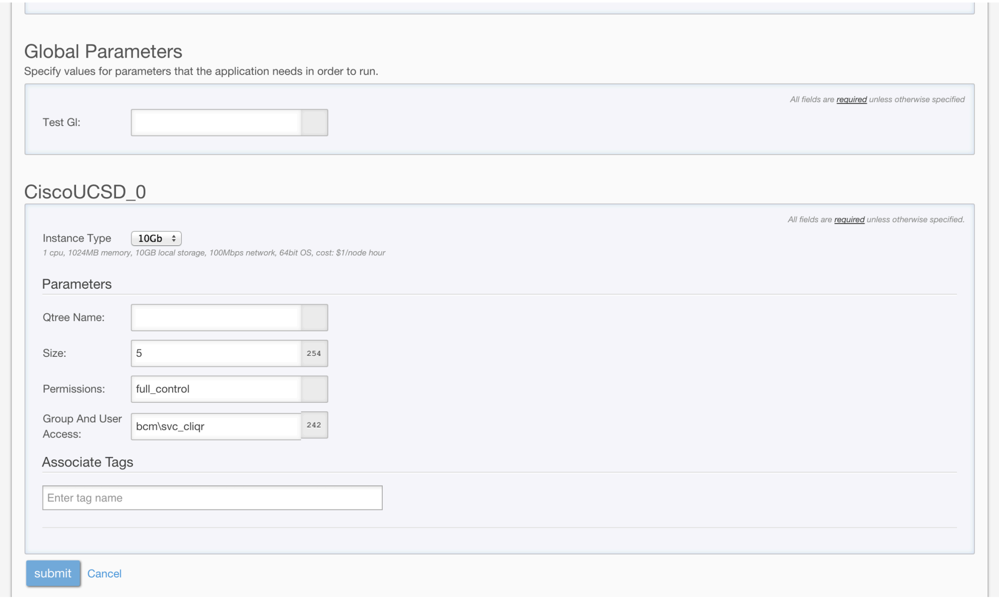
Set Workflow Details for Deployment Workflow

Workflow Name: create_cifs

values for parameters and/or set them up to accept input at the time of launching the application.

	Description	Source	Default Value	Validation Workflow
	Qtree Name	LaunchTimeInput		validate_qtree_name
	Size	LaunchTimeInput	5 GB	
	Permissions	LaunchTimeInput	full_control	No validation required
Jser_Access	Group and User Access	LaunchTimeInput	bcm\svc_cliqr	No validation required

9. Launch the UCSD application.



Global Parameters

Specify values for parameters that the application needs in order to run.

All fields are required unless otherwise specified

Test GI:

CiscoUCSD_0

All fields are required unless otherwise specified.

Instance Type: 10Gb

1 cpu, 1024MB memory, 10GB local storage, 100Mbps network, 64bit OS, cost: $1/node hour

Parameters

Qtree Name:

Size: 5 (254)

Permissions: full_control

Group And User Access: bcm\svc_cliqr (242)

Associate Tags

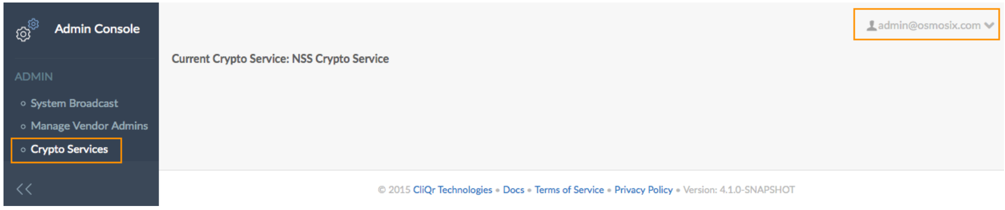
You have now configured and launched UCSD as a custom service.

- Validate Workflow
- Fetch Job-Associated VM Details
- Reconfigure Running Job
- Synchronize VM Cloud Information
- Stop Job
- Submit Job

CloudHSM Integration

- Overview
- Requirements
- Other References

CliQr supports AWS Cloud Hardware Security Module (CloudHSM), a hardware appliance that provides secure key storage and enables cryptographic operations within a tamper-resistant hardware module.


Admin Console
ADMIN
System Broadcast
Manage Vendor Admins
Crypto Services
<<
admin@osmosix.com
© 2015 CliQr Technologies • Docs • Terms of Service • Privacy Policy • Version: 4.1.0-SNAPSHOT

Current Crypto Service: NSS Crypto Service

To use CloudHSMs, the Log in as a SysAdmin must adhere to the following requirements:

- Configure the CCM for Luna Provider (lunaProvider.jar file) to ensure that you have copied this file to the CCM server in the */usr/local/tomcat/lib* directory so Tomcat can use this library. You will need to restart Tomcat so it automatically uses the lunaProvider library.
- Each tenant requires a unique encryption key.
- The CCM instance that interacts with the CloudHSM server must reside inside the same VPC as the CCM.
- Reboot the CCM before re-establishing the connection to the CloudHSM.

1. Safenet Luna WebHelp
2. AWS CloudHSM
3. AWS CloudHSM Getting Started Guide
4. AWS CloudHSM Forum
5. Connecting Multiple VPCs with EC2 Instances (SSL)

Configure SMTP Mail Servers in CCMs

- Overview
- The Mail Properties File
- SMTP Configuration Process

This section explains how to set up your SMTP mail server to use CCM to send emails.

⚠ CliQr does not support TLS ports. CliQr only supports SSL ports to configure SMTP mail servers.

ⓘ Add the password string in encrypted format.

The mail.properties file is available in your local Tomcat server at /usr/local/tomcat/webapps/ROOT/WEB-INF/mail.properties.

```
# The hostname or IP address of your SMTP server
# Currently mob-gen.com email domain is hosted by gmail
# Gmail requires smtp over ssl, do not modify these settings
mail.smtp.host=smtp.gmail.com
mail.smtp.auth=true
mail.smtp.port=465
mail.smtp.socketFactory.port=465
mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory
mail.smtp.socketFactory.fallback=false

# Email user to authenticate to gmail
mail.user.number=1

mail.user.1=<your_osmosix_email_addr>
mail.password.1=<your_email_password>
from.mail.user.1=<your_cliqrtech_email_addr>
from.mail.username.1=
```

To configure SMTP mail servers in CCM, follow this process.

1. Open the mail.properties file.
   vi /usr/local/tomcat/webapps/ROOT/WEB-INF/mail.properties
2. Modify the required settings in this file.
   For example, if using Gmail, the only lines to change are:

```
mail.user.1=<your_osmosix_email_addr>

mail.password.1=<your_email_password>

from.mail.user.1=<your_cliqrtech_email_addr>

from.mail.username.1=
```

Similarly, change the required lines in your mail.properties file to ensure that you can access you mail from the CCM

3. Restart Tomcat.
   /etc/init.d/tomcat restart

Deployment Workflow Callout Scripts

At various stages in the deployment lifecycle of VMs, CliQr supports the ability to control the behavior of the provisioning process. The different lifecycle points where the behavior can be controlled are called *topics*. The behavior is controlled by scripts via *callouts* that are assigned to topics. A common use case for callouts is to query an IPAM tool during the IP Address Allocation (IPAM) topic to get an IP address and during the IP shutdown topic (IPAM2) to de-allocate the IP address. See the Infoblox integration page for an example of implementing this use-case.

Keep in mind that callouts are configured on a per-CCO basis and will apply to every VM provisioned from that CCO. If different behaviors are required, control logic (if/then/case) can be used inside the callout script.

Each callout consists of two key parts: a configuration file (callout.conf) and the script to be executed. These files are placed in the /usr/local/osmosix/callout/<name> path on the CCO. The name of the sub-folder that you use is arbitrary, but a best-practice is to use the name of the topic that the callout is for. Ex: /usr/local/osmosix/callout/ipam/<files>

Each callout script, when executed from CliQr, has access to a wide variety of environment variables that can be used, including cloud type, deployment environment, and so forth.

A full list is available in the callout script log at /usr/local/osmosix/callout/<name>/logs/<timestamp>.

The callout scripts use the same parameters and incoming variables. Each script exposes a different variable and are mutually exclusive. You can execute any script when required.

Each of these scripts are explained in the sections below.

Use the table of contents above to link directly to each script explanation.

Script Topic	Description	Supported Clouds
vmNaming	Called before each node is launched This script is provided (injected into the script) all the name variables (name of application, name of tier, image selected,) for each job.	See VM (Node) Name Config for additional details on the supported callouts for the supported clouds.

ipam	Network and OS-specific configuration	VMware
preDestroy	Called just before a node is destroyed	VMware ⚠ Effective CloudCenter 4.5.1, preDestroy is not available for AWS and OpenStack. See Manage Clouds > Actions at the Cloud Region Level for alternate choices.
ipam2 Also known as ***ipamDealloc***	Called just after a node is destroyed	VMware ⚠ Effective CloudCenter 4.5.1, ipamDealloc is supported for AWS and OpenStack.

callout.conf Supported Attributes

This callout script supports standard Java property file format, using <key>=<value>, each on a separate line.

Key	**Description**	**Required?**
name	The module name	Yes
type	The module type CloudCenter currently supports the EXEC (shell script) type.	Yes
topic	The module topic is required and important: • vmNaming • ipam • preDestroy • ipam2	Yes
executable	The name of the script binary executable (executable=run.sh) residing in the same directory (the module path). The callout script is invoked with instance properties setup as environment variables, each variable name is constructed with the prefix eNV_ string plus the original variable name.	Yes
debug	Optional. The debug mode: • true • false (default)	No
disabled	Optional. The module is disabled if set to true, which is useful for development/test: • true • false (default)	No
reinject	Optional. Reinjects the callout script output properties back to the input environment variables of subsequent callouts, so the subsequent callout can pick up the value: • true • false (default) CliQr passes the parameters and the environment variables listed above to each module. If you set reinject=true, all properties for previously-used callout scripts are also re-injected into the additional callout modules, including arbitrary additional key/values that you wish to specify.	No

Environment Variables for Callout Scripts

The Callout script accepts environment variables as input parameters. The list of variables depends on the node type. The following table provides a sample:

Variable	**Sample Values or Type**
eNV_osName	Linux, Windows
eNV_vmName	A string passed by the vmNaming module or auto-generated by CliQr

eNV_JOB_ID	An integer to identify the application VM (only)
eNV_launchUserId	An integer to identify the User ID of the person launching the script/module
eNV_launchUserName	A string to identify the user name of the person launching the script/module

All job application settings for application VMs are also available as eNV variables. See CliQr-Defined Parameters for additional context.

Best Practice
Turn on the debug level and check the debug logs (see Locate Log Files) to view a list of all available input variables.

Configure Each Callout Script

Configure each script separately in a callout.conf file.

You can configure the each of these callout scripts at the region level, ***not at the tenant level***, on a per-CCO basis. The following example depicts the configuration procedure to add the vmNaming callout script.

To add a callout script, follow this process.

1. Create the following directory on the CCO:

   `/usr/local/osmosix/callout/vmname/`

2. Create the following file in this directory:

   `/usr/local/osmosix/callout/vmname/callout.conf`

3. Create a file for the script:

   `/usr/local/osmosix/callout/vmname/<script name>`

4. Ensure to execute permissions:

   `chmod 777 <script>`

5. Reference this file in the callout.conf file.

The supported environment variables for the vmNaming script:

Variable	Sample value or type
eNV_JOB_ID	integer (application VM only)
eNV_launchUserId	integer
eNV_launchUserName	string

The supported CliQr key for the vmNaming script:

CliQr-Required Key	Description
vmName	Name of the VM

A sample vmNaming callout script output:

run.sh

```
#!/bin/bash

echo "vmName=`uuidgen`"
```

This script allows you to change the name of the VM. See VM (Node) Name Config to rename the VM using the CCM UI for OpenStack, VMware, and Google clouds.

As part of the integration, create a IPAM module and include the dynamically-invoked callout script when launching the CCO. The module can be dynamically loaded/reloaded (auto-load) or loaded at CCO start-up time. By default, auto-load is disabled.

The IPAM module's callout script includes (but is not restricted to) the following parameters:

- DNS server list
- DNS suffix list
- Number of vNIC
- Number of vNIC's IP address
- Numbers of vNIC's netmask
- VM name

Once the script is executed, all deployments for that cloud discover IP addresses managed by the IPAM module.

> ⚠ The callout script option for IPAM integrations is only available for VMware clouds.

The callout script path is /usr/local/osmosix/callout, where each module is a sub-folder under the script path.

Example

```
UserClusterName="cluster01"

eNV_Cloud_Setting_UserDataCenterName="dc02"

eNV_NumTasks="1" eNV_UseBatchTaskList="0"

eNV_Cloud_Setting_UserResourcePoolName="resourcepool1"

eNV_Cloud_Setting_UserClusterName="cluster01"
```

Supported Properties

The multiple key-value pair that is output for each callout script.

Key	Description	Required?
domainName	Linux domain name	Yes
hwClockUTC	H/W clock UTC	Yes
timeZone	Time zone	Yes
osHostname	host name	Yes
DnsServerList	DNS server list (comma separated)	
DnsSuffixList	DNS Suffix list (comma separated)	
nicCount	The number of vNIC	Yes
nicIP_n	The number of n vNIC's IP address	Yes
nicNetmask_n	The number of n vNIC's netmask	Yes
nicGateway_n	The number of n vNIC's gateway (CCO) IP address	
nicDnsServerList_n	The number of n vNIC's DNS server list (comma separated)	Yes
nicUseDhcp_0	As part of the IPAM script, provide dummy values for nicIP_n and nicDnsServerList_n. However, these values are overwritten by the DHCP settings.	Yes, if using IPAM callout and the addressing is assigned to use DHCP.
ANY	This property is supported if the `reinject` setting is true Example: myCustomParam=myValue	

Windows-Specific IPAM Properties

Key	Description	Required?

timeZoneId	The Windows Index ID for this time zone.	Yes
fullName	The name of the Admin user	Yes
organization	The name of the organization (string)	Yes
productKey	The Windows product key	Yes
setAdminPassword	The Admin password	Yes
changeSid	A true or false value for the Microsoft SID	Yes
deleteAccounts	A true or false value.	Yes
dynamicPropertyName	Reserved name holder for arbitrary property	Yes
dynamicPropertyValue	Reserved value holder for arbitrary property	Yes
custSpec	The Guest Customization Specification in VMware that you want to use on the deployment callouts page.	No
domain	Used for automatically joining a domain	If any domain value is missing, the workgroup key is required. If all three domain values are present, then workgroup is not required.
domainAdminName	Used for automatically joining a domain	
domainAdminPassword	Used for automatically joining a domain	
workgroup	The workgroup in which to place the VM.	

Windows-Specific Example

run.sh

```
#!/bin/bash

echo "setAdminPassword=abcd"
echo "timeZoneId=10 *"
echo "fullName=Enterprise ABCD"
echo "organization=ABCD"
echo "productKey=..."
echo "changeSid=true"
```

Sample IPAM Callout Script

run.sh

```
#!/bin/bash

echo “DnsServerList=8.8.8.8,10.0.0.100"

echo "nicCount=2"
echo "nicIP_0=10.0.0.100"
echo "nicDnsServerList_0=1.2.3.4,5.6.7.8"
echo "nicGateway_0=10.0.0.1"
echo "nicNetmask_0=255.255.255.0"

echo "nicIP_1=192.1.0.100"
echo "nicDnsServerList_1=10.10.10.10"
echo "nicGateway_1=192.1.0.1"
echo "nicNetmask_1=255.255.255.0"

echo "domainName=test.org"
echo "hwClockUTC=true"
echo "timeZone=Canada/Eastern"
echo "osHostname=testhost1”
```

For AWS and OpenStack Configurations
Effective CloudCenter 4.5.1, the following parameters are not supported for AWS and OpenStack:

- `domainName`
- `hwClockUTC`
- `timeZone`
- `osHostname`

Additionally, the nicCount parameter only allows one NIC to be used at any time.

If the VM configuration includes multiple NICs, then the callout script will be called once for each NIC.

The preDestroy script allows you to notify the system just *before* you de-provision the network configuration.

CliQr does not look for any output from this script as it is just a notification.

The IPAM2 script allows you to cleanup your environment and only works with custom property supported by `reinject` setting.

IPAM2 example:

run.sh

```
 #!/bin/bash

./delete_record_by_ip.sh $IP
```

CliQr does not look for any output from this script as it is just a notification.

Sample IPAM Callout

```
 #!/usr/bin/env python
import infoblox, sys, requests, os, random
requests.packages.urllib3.disable_warnings()

#Assign command line arguments to named variables
hostname = os.environ['vmName']
domain = "vm.cliqr.com"
fqdn = hostname + "." + domain
network = "10.49.18.0/23" #sys.argv[2]
netmask = "255.255.254.0"
gateway = "10.49.19.254"
dns_server = "10.48.112.33,10.52.112.19"

#Setup connection object for Infoblox
iba_api = infoblox.Infoblox('10.49.9.163', 'admin', 'infoblox', '1.4',
iba_dns_view='VM-view', iba_network_view='default', iba_verify_ssl=False)

try:
 #Create new host record with supplied network and fqdn arguments
 ip = iba_api.create_host_record(network, fqdn)
 print "DnsServerList="+dns_server
 print "nicCount=1"
 print "nicIP_0=" + ip
 print "nicDnsServerList_0="+dns_server
 print "nicGateway_0="+gateway
 print "nicNetmask_0="+netmask
 print "domainName="+domain
 print "HWClockUTC=true"
 print "timeZone=Canada/Eastern"
 print "osHostname="+hostname
 print "infobloxFQDN="+fqdn
except Exception as e:
 print e
```

Infoblox Integration

CliQr supports IP Address Management (IPAM) integration to manage IP address for deployments.

This section provides information on integration with Infoblox, an IPAM provider, by executing multiple callout scripts on the CCO. See Callout Scripts for additional information on each callout script.

See the CliQr InfloBlox Integration video for a short demonstration of how CliQr supports Infoblox for IP address and DNS name assignment.

This integration is only available for VMware and OpenStack (effective CloudCenter 4.5.1) clouds.

1. Contact CliQr Support to obtain the module templates and save it to **/usr/local/osmosix/callout**.
2. Make changes to callout scripts according to your test environment.
3. Model a sleep job, add some environment variables, run the job and check the callout log, and verify that the variables are exported correctly.
4. Model a multi-tier web app, add some environment variables, run the job, check the callout log, and verify that the variables are exported correctly.

Use the Infoblox-API-Python module to integrate with Infoblox.

The following example displays a script that uses the Infoblox-API-Python module. This script **requires python-requests 2.5** and can be called directly from the callout:

createHost.py

```
#!/usr/bin/env python
import infoblox, sys
#Check to see if command line included enough arguments.
if (len(sys.argv) < 3):
 print "Usage: createHost.py <fqdn> <network CIDR>"
 quit()
#Assign command line arguments to named variables
fqdn = sys.argv[1]
network = sys.argv[2]
#Setup connection object for Infoblox
iba_api = infoblox.Infoblox('10.110.1.45', 'admin', 'infoblox', '1.6', 'default',
'default', False)
try:
 #Create new host record with supplied network and fqdn arguments
    ip = iba_api.create_host_record(network, fqdn)
    print "nicCount=1"
    print "nicIP_1=" + ip
except Exception as e:
    print e
```

This is an example of integrating callout with the Infoblox application.

callout.conf

```
name=infoblox
type=exec
topic=ipam
debug=true
executable=createHost.py
reinject=true
disabled=false
```

Jenkins Integration

For pre-modeled Jenkins projects (for example, Maven, to fetch the source code from Git/SVN), you can integrate with CloudCenter using the CliQr Jenkins plugin.

Effective CloudCenter 4.2, you do not need to manually copy this file, CloudCenter provides a download URL to make this plugin available to Jenkins users. Contact CliQr Support to obtain the download location.

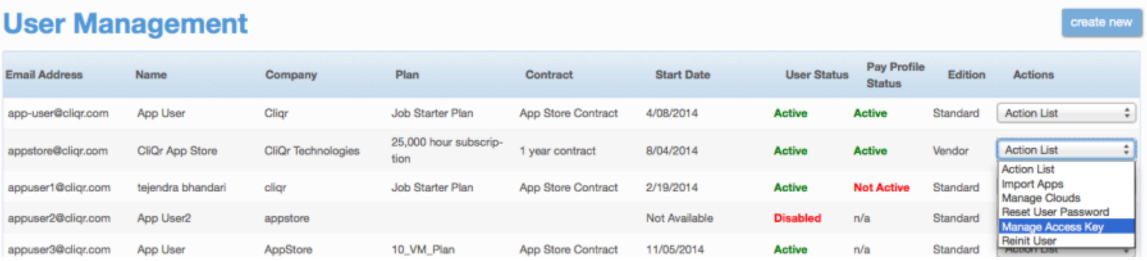
The CliQr Jenkins plugin provides complete integration between Jenkins and CliQr CloudCenter by allowing users to directly launch deployments on any CliQr-Supported Cloud from a Jenkins server.

Additionally, users can upgrade an existing deployment by specifying upgrade scripts for each tier.

- If you are new to Jenkins, setup a maven project on Jenkins with Github as its source repository. See http://www.youtube.com/watch?v=lTQGi5jzjvo for additional details.
- The supported Jenkins versions value must be **>=1.624** to use the CliQr Jenkins plugin.
- The required Java version for the Jenkins server must be **Java 7**.

To install the CliQr Jenkins plugin, follow this procedure:

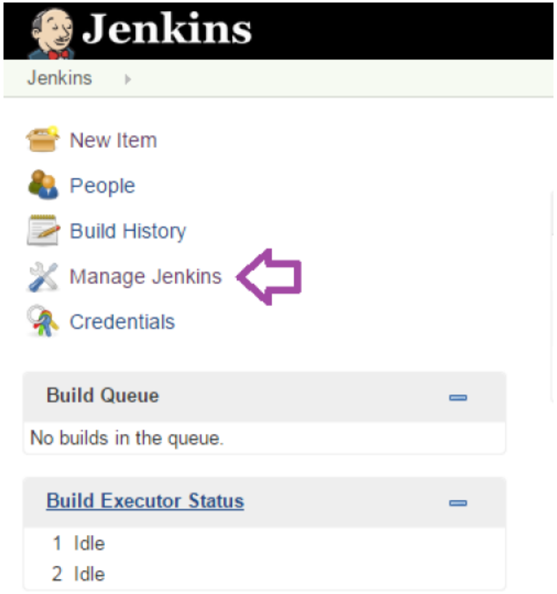
1. Contact CliQr Support to obtain the download link for the CliQr Jenkins plugin.
2. Log into CCM using your admin credentials.
3. Generate the API Management (Access) Key for the Jenkins user.



User Management

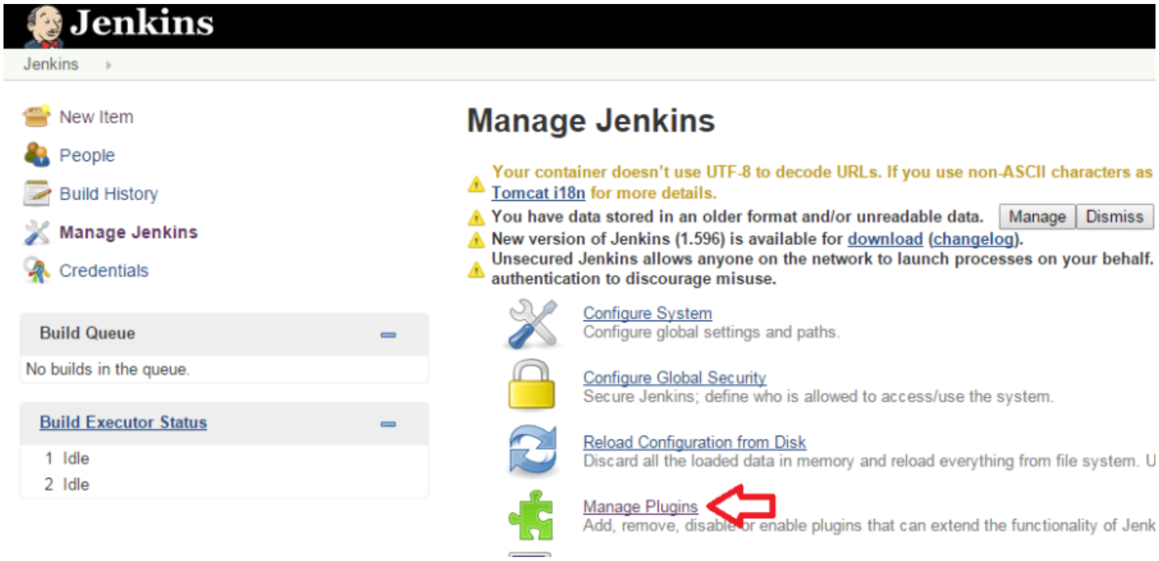
Email Address	Name	Company	Plan	Contract	Start Date	User Status	Pay Profile Status	Edition	Actions
app-user@cliqr.com	App User	Cliqr	Job Starter Plan	App Store Contract	4/08/2014	Active	Active	Standard	Action List
appstore@cliqr.com	CliQr App Store	CliQr Technologies	25,000 hour subscription	1 year contract	8/04/2014	Active	Active	Vendor	Action List
appuser1@cliqr.com	tejendra bhandari	cliqr	Job Starter Plan	App Store Contract	2/19/2014	Active	Not Active	Standard	
appuser2@cliqr.com	App User2	appstore			Not Available	Disabled	n/a	Standard	
appuser3@cliqr.com	App User	AppStore	10_VM_Plan	App Store Contract	11/05/2014	Active	n/a	Standard	Action List

4. Model the Application so this user can access artifacts from the Jenkins build server.
5. In Jenkins, go to **Manage Jenkins** > **Manage Plugins** > **Advanced** > **Upload Plugin** to upload and install the CliQr Jenkins Plugin.



No builds in the queue.

1 Idle
2 Idle



No builds in the queue.

1 Idle
2 Idle

Manage Jenkins

Your container doesn't use UTF-8 to decode URLs. If you use non-ASCII characters as Tomcat i18n for more details.

You have data stored in an older format and/or unreadable data.

New version of Jenkins (1.596) is available for download (changelog).

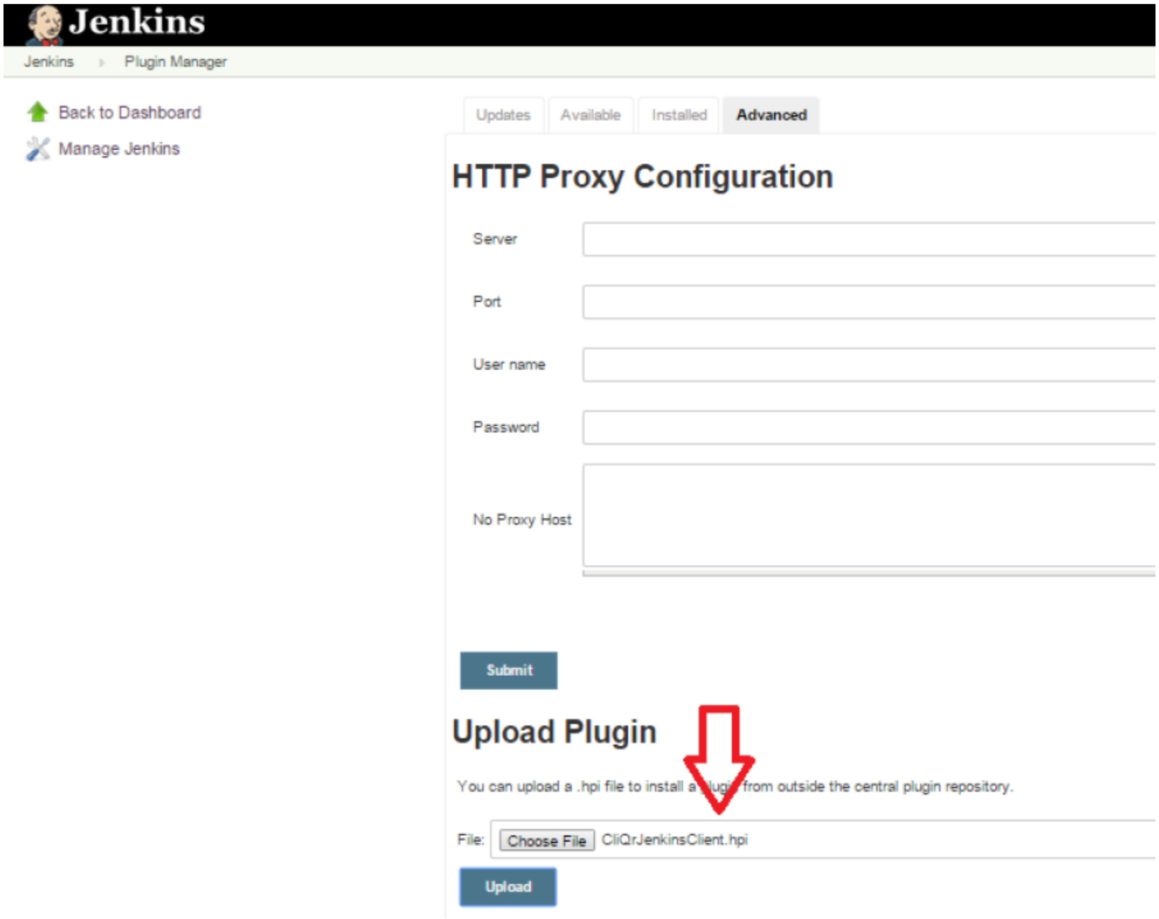
Unsecured Jenkins allows anyone on the network to launch processes on your behalf. authentication to discourage misuse.

Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. U

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenk



HTTP Proxy Configuration

Server

Port

User name

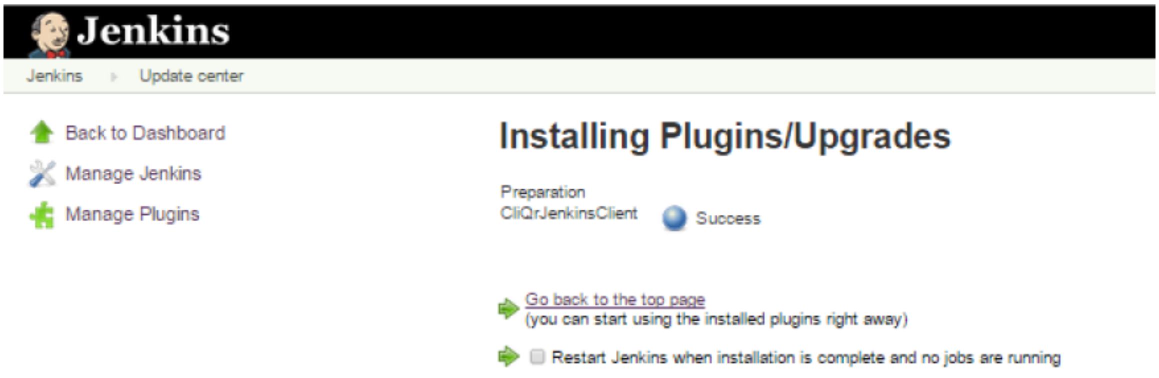
Password

No Proxy Host

Upload Plugin

You can upload a .hpi file to install a plugin from outside the central plugin repository.

File: CliQrJenkinsClient.hpi



Installing Plugins/Upgrades

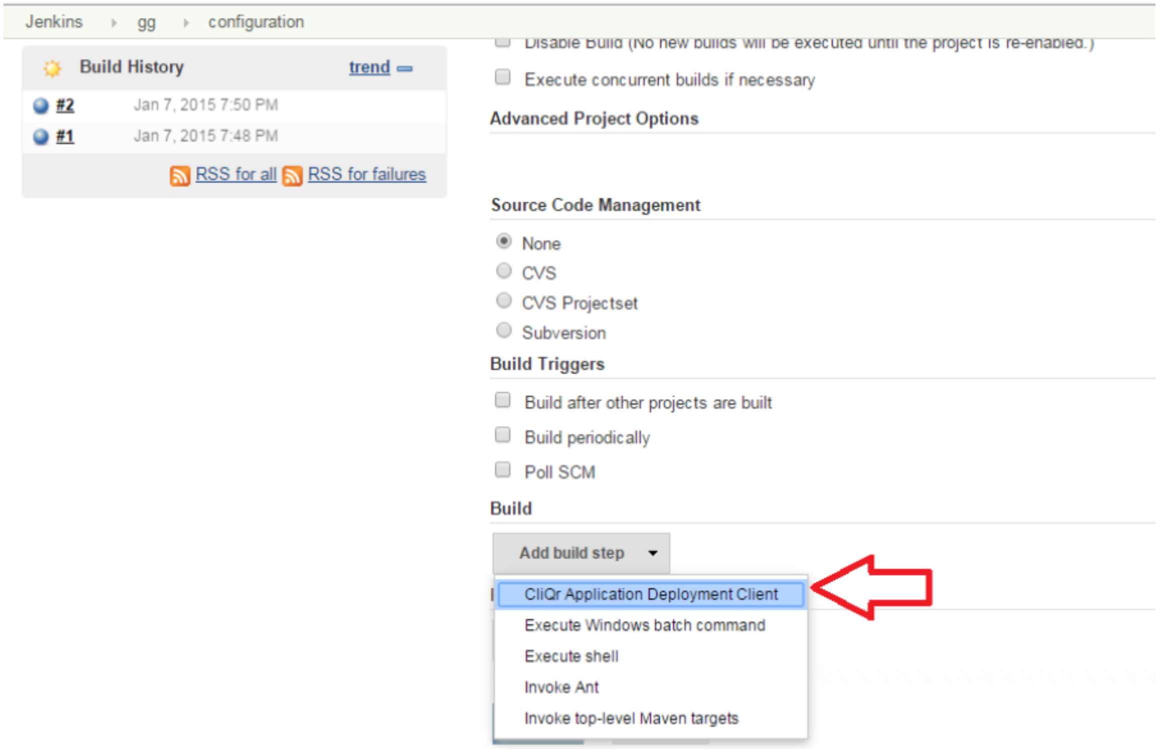
Preparation

CliQrJenkinsClient ● Success

Go back to the top page
(you can start using the installed plugins right away)

□ Restart Jenkins when installation is complete and no jobs are running

6. After you install theCliQrJenkinsPlugin, go to your existing/new project to configure post-build step and fetch the source code from the Git/SVN using a Maven project into the Jenkins Build.


Jenkins
gg
configuration
trend
RSS for all
RSS for failures
Add build step
CliQr Application Deployment Client
Execute Windows batch command
Execute shell
Invoke Ant
Invoke top-level Maven targets

Build History

#2	Jan 7, 2015 7:50 PM
#1	Jan 7, 2015 7:48 PM

□ Disable Build (No new builds will be executed until the project is re-enabled.)

□ Execute concurrent builds if necessary

Advanced Project Options

Source Code Management

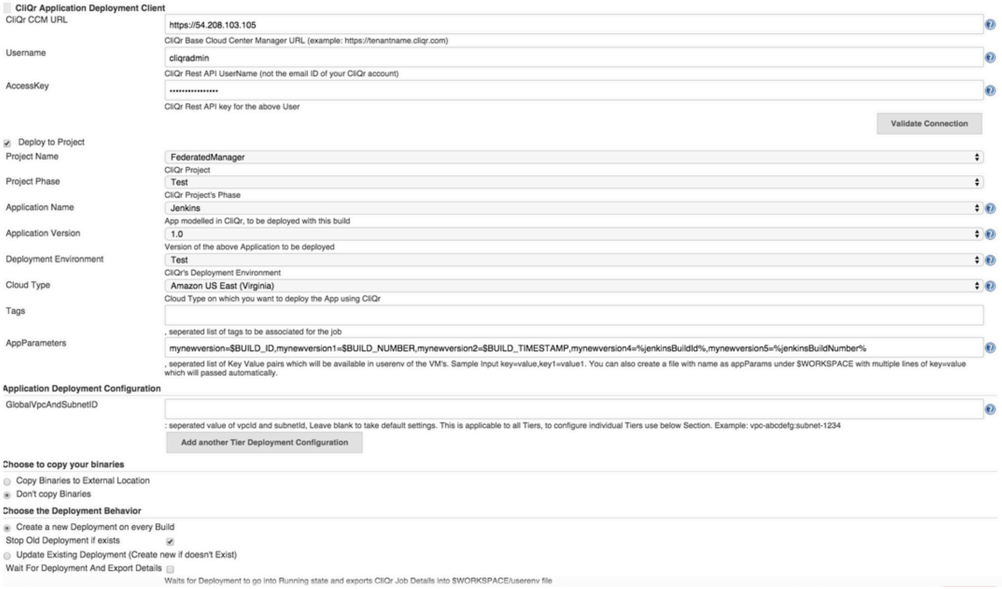
● None
○ CVS
○ CVS Projectset
○ Subversion

Build Triggers

□ Build after other projects are built
□ Build periodically
□ Poll SCM

Build

7. Configure the CliQr Application Deployment Client in Jenkins for continuous integration from build system and deployment (new or upgrade on an existing node).



CliQr Application Deployment Client

CliQr CCM URL: https://54.208.103.105
CliQr Base Cloud Center Manager URL (example: https://tenantname.cliqr.com)

Username: cliqradmin
CliQr Rest API UserName (not the email ID of your CliQr account)

AccessKey: ••••••••••••••••
CliQr Rest API key for the above User

☑ Deploy to Project

Project Name: FederatedManager
CliQr Project

Project Phase: Test
CliQr Project's Phase

Application Name: Jenkins
App modelled in CliQr, to be deployed with this build

Application Version: 1.0
Version of the above Application to be deployed

Deployment Environment: Test
CliQr's Deployment Environment

Cloud Type: Amazon US East (Virginia)
Cloud Type on which you want to deploy the App using CliQr

Tags:
, seperated list of tags to be associated for the job

AppParameters: mynewversion=$BUILD_ID,mynewversion1=$BUILD_NUMBER,mynewversion2=$BUILD_TIMESTAMP,mynewversion4=%jenkinsBuildId%,mynewversion5=%jenkinsBuildNumber%
, seperated list of Key Value pairs which will be available in userenv of the VM's. Sample Input key=value,key1=value1. You can also create a file with name as appParams under $WORKSPACE with multiple lines of key=value which will passed automatically.

Application Deployment Configuration

GlobalVpcAndSubnetID:
: seperated value of vpcId and subnetId, Leave blank to take default settings. This is applicable to all Tiers, to configure individual Tiers use below Section. Example: vpc-abcdefg:subnet-1234

Choose to copy your binaries

○ Copy Binaries to External Location
● Don't copy Binaries

Choose the Deployment Behavior

● Create a new Deployment on every Build
Stop Old Deployment if exists ☑
○ Update Existing Deployment (Create new if doesn't Exist)
Wait For Deployment And Export Details □
Waits for Deployment to go into Running state and exports CliQr Job Details into $WORKSPACE/userenv file

The parameters for the CliQr Application Deployment Client page are listed in the following table:

Parameter	Description
CliQr CCM URL	The the IP address of the CCM server. Verify that trusted certificates (see Certificate Authentication for additional context), are added to the CCM server (*not* self-signed certificates).
Username	Username listed in the Manage Access Key section (see API Authentication for additional context).
AccessKey	The API Management Key for the user listed in the he Manage Access Key section (see API Management Key for additional context).
Deploy to Project	Use this flag to deploy to a project, instead of a general deployment environment (see Manage Projects and Phases for additional context).

Project Name	If you select the **Deploy to Project** flag, this parameter is displayed and the list of projects are fetched from the CCM server. Based on this parameter value, only applications associated with this project are filtered out in the **Application Version** field.
Project Phase	Based on the project name, a dropdown list of all Phases in that project are displayed. Use this value in the Deployment Environment field to filter the project.
Application Name	From the dropdown list of applications listed in the CCM UI, select the required application to deploy. After entering your credentials, be prepare to wait for some time as the Application Management APIs APIs may take a while to load.
Application Version	Based on your application, select the application version from this dropdown list.
Deployment Environment	Select the required deployment environment to deploy your application. Be sure to verify and check all default settings like default cloud, default instance type, and so forth as CloudCenter uses these default settings for each deployment.
Cloud Type	Select one cloud type from this dropdown list of cloud types that are present in your deployment environment.
AppParameters	A comma separated list of key-value pairs to pass as global parameters. For example: abcd=wow, cdef=cliqrRocks CloudCenter includes the $BUILD_ID, $BUILD_NUMBER, $BUILD_TAG, $JOB_NAME, and if available, $BUILD_TIMESTAMP from other plugins. You can add a variable in this section to fetch parameters that are shared by other jobs. For example: abc=${BUILD_PARENT_NUMBER} or abc=$BUILD_PARENT_NUMBER You also have the option to retrieve parameters from the $WORKSPACE/appParams file that contains multiple lines of key-value parameter pairs. You can then uses these parameters to pass passwords or other sensitive information without displaying them in the CCM UI.
Binaries to be Copied	A comma separated list of file or folder paths that needs to be copied from the Artifact Repository, an external repository, or an external host.
Copy Binaries to External Location	• External Host: Your external host / repository IP or public DNS • HostUsername: The SSH login username with which we can SCP to this host • Password: The password for the above user with which authentication happens. • Target Folder on Above Host: The target location on the External Host where the files or folders are copied. If /tmp/app1 is the given location, all your binaries will be available under /tmp/app1/latest
Create a new Deployment on every Build	You can only do this once for deployments. This option creates a brand new deployment for every build.
Update an Existing Deployment	You can only do this once for deployments. • Updates a previous deployment that was launched from the CliQr Jenkins plugin during a previous build for the same project. • If the previous deployment is still in progress (job) and is not yet in the Running state, the CliQr Jenkins plugin waits till the deployment is in the running state before triggering an update (see Deployment and VM States for additional context). • If the previous deployments ends up as an error or if that deployment is stopped or cancelled from the the CCM server, the the CliQr Jenkins plugin launches a new deployment as part of the Update process. • If this is the first build, the CliQr Jenkins plugin creates a new Deployment and from the next successful build it uses the existing deployment. • UpdateScripts: A comma separated list of tierName**:**scriptToExecute scripts that are executed in the order mentioned here. For example: AppCluster:/shared/app/petclinic/update.sh,Database:/shared/app/updatemysql.sh, AppCluster:/shared/app/startServer.sh

In Update Scripts, $BUILD_ID or %jenkinsBuildId% can be passed as an argument to point the *Binaries to be Copied* during an update deployment.

The %jenkinsBuildId% macro is not an applications-specific macro. This CliQr-defined macro applies to deployments that are launched using the Jenkins plugin.

The jenkinsBuildId macro is mainly used to pass the Jenkins Build ID to the userenv of the app deployment. Any deployment triggered by the Jenkins plugin will automatically have jenkinsBuildId in the userenv and will be used to point to the right binaries in repo/storage. For example, if a web server has a previous war file path set to /shared/app/petclinic/latest/, then this war file (petclinic.war) can now use this macro to point to /shared/app/petclinic/%jenkinsBuildId%/petclinic.war.

In update deployment scenarios, the jenkinsBuildId macro changes the value that should be passed to existing deployments as userenv has old the jenkinsBuildId value during the deployment.

The folder name that CliQr creates in the target location will now use the jenkinsBuildId value instead of the random timestamp value.

This option creates a Brand new deployment on every build.

When you update an existing deployment:

- It updates a previous deployment that is launched from the Jenkins plugin during the previous builds of the same project.
- If the previous deployment job is still in progress and is not in the Running state, the plugin waits till it enters the running state and then triggers an update.
- If the previous deployments ends in an error or if that deployment is stopped/cancelled from the CCM, the plugin launches a fresh deployment as part of update.
- If it is the first build, this plugin creates a new deployment and for the next successful build it uses the existing deployment.

SSO Integrations

SSO AD

Overview

Some enterprises have their own Active Directory (AD) or other similar setup and prefer to use those credentials to login into the external applications and platforms. CloudCenter does not support direct AD authentication, and instead supports integration using Single Sign-On (SSO) between the CloudCenter as a Service Provider (SP) and a customer's Identity Provider (IDP) such as ADFS.

CloudCenter supports a multi-tenant model where each vendor is modeled as a tenant. The tenants have a single root hierarchical tree structure. Each tenant has its' own set of users. One of the users is a tenant admin (also referred to as the root admin or platform admin) that has special administrative permissions.

CloudCenter does not authenticate directly to LDAP or AD.

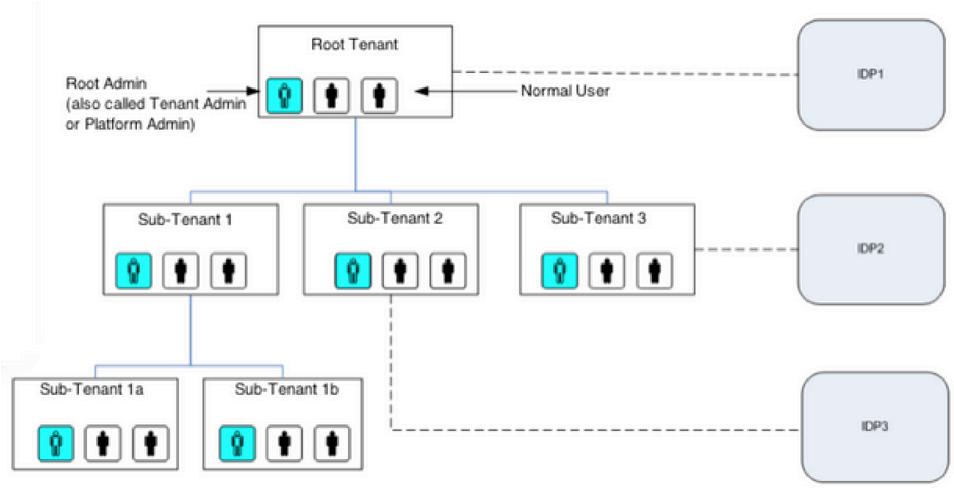
CloudCenter only interacts with LDAP/AD through a SSO IDentity Provider (IDP) that supports SAML 2.0 protocol (for example, Ping Identity, ADFS, Shibboleth, and so forth).

To implement SSO using CloudCenter:

1. You must then configure the CCM to re-direct the authentication to the SSO IDP.
2. You must also map some additional user custom properties (returned by the SAML IDP) to the user activation profile.
3. Once you complete all these steps successfully, CloudCenter automatically assigns the proper user group membership and additional roles and permissions.

User Authentication

Despite a user (User X) being authenticated by an external Identify Provider (IDP), User X also requires a corresponding presence in the CCM VM's user database. In the SSO environment, after User X is authenticated by IDP and uses the CCM VM for the first time, a User X authentication is automatically created in the CCM user database as long as the platform admin has created the tenants and tenant admins.



Each tenant can point to it's own SSO:

- You can configure each CliQr tenant to have a dedicated alias hostname and use an external IDP to authenticate its users.
- Each CliQr tenant and user has an *externalId* to associate with an external organization and user.

Handling Deleted Users

If you delete a user from the IDP database, the deleted user cannot log into CloudCenter but any configuration and associated dependencies continue to remain in the CloudCenter platform.

SAML SSO

- Overview
- CliQr Support
- CliQr SAML Authentication Configuration
- Other References

Overview

✓ **CloudCenter does not authenticate directly to LDAP or AD.**

CloudCenter only interacts with LDAP/AD through a SSO IDentity Provider (IDP) that supports SAML 2.0 protocol (for example, Ping Identity, ADFS, Shibboleth, and so forth).

To implement SSO using CloudCenter:

1. You must then configure the CCM to re-direct the authentication to the SSO IDP.
2. You must also map some additional user custom properties (returned by the SAML IDP) to the user activation profile.
3. Once you complete all these steps successfully, CloudCenter automatically assigns the proper user group membership and additional roles and permissions.

A CCM instance supports Security Assertion Markup Language (SAML) 2.0 SSO through Spring Security SAML Extension.

The SysAdmin can set up SAML integration at the root level or the tenant level. To accurately configure this integration, you must have the following information for the root tenant or sub-tenant (as applicable to your deployment):

- Tenant configuration information
- Activation profile information

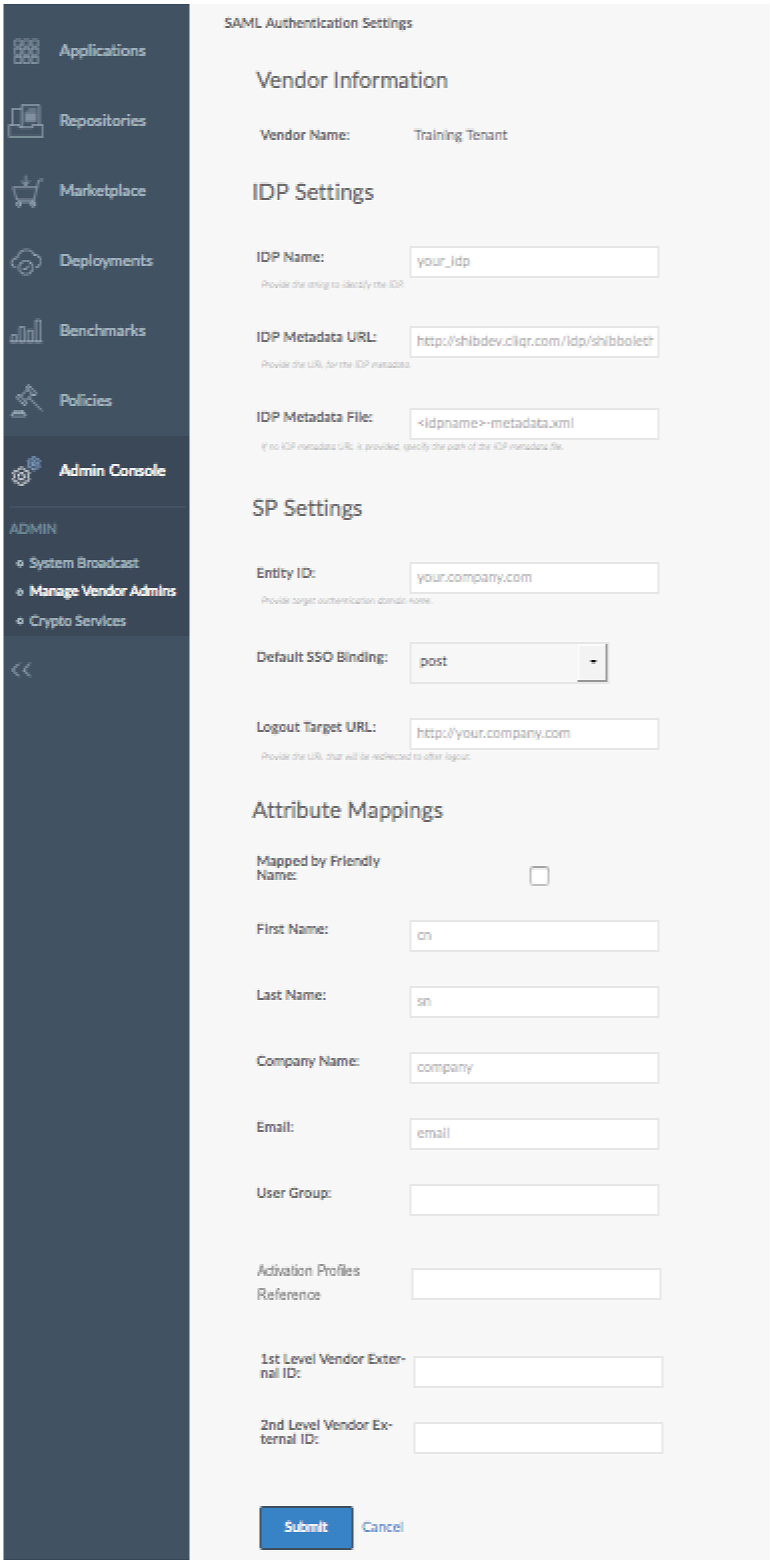
CliQr Support

Contact CliQr Support for additional information.

CliQr SAML Authentication Configuration

To configure a tenant to use SSO, follow this procedure:

1. Create a tenant (see Add Sub-Tenants)
    1. Short Name – give a string without white spaces and special characters.
    2. External Id – enter the ID of the organization in the external system with which the CliQr tenant is associated.
    3. Tenant – the CCM server domain name alias for the tenant. This will serve as the end point of the Service Provider (SP) from the SSO perspective. See Add Sub-Tenants for additional context.
2. Login as the newly created tenant admin and create an Activation Profile (see Activation Profiles).
3. Click the **Tenant Info** tab and select the newly created activation profile as Default Activation Profile.
4. Gather the IDP information to see what Subject Attribute the IDP will provide and create a IDP Subject Attribute to CliQr User Property mapping plan.
5. Login as SysAdmin, click the **Manage Vendor Admins** tab and select the **Authentication Settings** action item.
6. Enter the information in the IDP Settings, SP Settings and Attribute Mappings sections and click Submit.



SAML Authentication Settings

Vendor Information

Vendor Name: Training Tenant

IDP Settings

IDP Name: your_idp
Provide the string to identify the IDP.

IDP Metadata URL: http://shibdev.cliqr.com/idp/shibboleth
Provide the URL for the IDP metadata.

IDP Metadata File: <idpname>-metadata.xml
If no IDP metadata URL is provided, specify the path of the IDP metadata file.

SP Settings

Entity ID: your.company.com
Provide target authentication domain name.

Default SSO Binding: post

Logout Target URL: http://your.company.com
Provide the URL that will be redirected to after logout.

Attribute Mappings

Mapped by Friendly Name: □

First Name: cn

Last Name: sn

Company Name: company

Email: email

User Group:

Activation Profiles Reference

1st Level Vendor External ID:

2nd Level Vendor External ID:

Submit Cancel

- **IDP Metadata**: To establish the mutual trust between CliQr and the IDP.
- **Entity ID**: The target domain name for this authentication
- **Logout URL**: If you are logging into your company's SAML page, you must specify the URL of the page that you want the logged in users to be redirected to when they log out of the SAML page.
- **Attribute Mapping Section**: These are the fields from the IDP that will be mapped to user attributes within CliQr. If you are unsure about these fields, please contact your IDP administrator. At a minimum, you need to provide the first name, last name, email address, and external User ID.
- **User Group:** Specify the field name from the IDP that will be used to determine the group to which the user belongs.
- **Activation Profiles Reference**: Identify an attribute in your metadata to pick an associated activation profile instead of the default profile.

7. Download the SP Metadata and send it to the IDP administrator to register the SP.

Other References

See the Ping Identity knowledge Base for an example of mapping IDP groups to SP groups.

Use Case: Shibboleth SSO

- CliQr Support
- Install Shibboleth
- Configure Shibboleth

CliQr Support

Contact CliQr Support for additional information.

> ✓ **CloudCenter does not authenticate directly to LDAP or AD.**
>
> CloudCenter only interacts with LDAP/AD through a SSO IDentity Provider (IDP) that supports SAML 2.0 protocol (for example, Ping Identity, ADFS, Shibboleth, and so forth).
>
> To implement SSO using CloudCenter:
>
> 1. You must then configure the CCM to re-direct the authentication to the SSO IDP.
> 2. You must also map some additional user custom properties (returned by the SAML IDP) to the user activation profile.
> 3. Once you complete all these steps successfully, CloudCenter automatically assigns the proper user group membership and additional roles and permissions.

Install Shibboleth

1. Prepare Ubuntu base image with Tomcat 6.

   > ⚠ The following instructions are for Tomcat6 (as Tomcat 7 has limitations).
   >
   > You can also use Jetty 7, however, the following instruction will differ if you use Jetty.

2. Download the latest Shibboleth IDP software from here http://shibboleth.net/downloads/identity-provider/latest/ to /shib-distro/.
3. Unzip the archive **:**
   sudo unzip /opt/shib-disto/shibboleth.zip
4. Copy files from the endorsed directory to Tomcat.
   sudo cp /shib-disto/shibboleth/endorsed/* /usr/local/tomcat6/endorsed/".
   You may need to make this the endorsed directory:
   sudo mkdir /usr/local/tomcat6/endorsed
5. Download tomcat6-dta-ssl-1.0.0.jar and copy to TOMCAT_HOME/lib:
   sudo cp /tmp/tomcat6-dta-ssl-1.0.0.jar /usr/local/tomcat6/lib
6. Install Shibboleth:
   a. **cd /shib-distro/shibboleth/**
   b. **sudo ./install.sh**
   c. Press **Enter** to accept default install path of /opt/shibboleth-idp.
   d. Press **Enter** to accept the Fully Qualified Domain Name (**FQDN**) of server (for example, idp01.cliqr.com).
   e. Enter a password to create the keystone.
7. Turn Off Certificate checking for Attribute requests (see Disabling Web Server CA Validation for Attribute Requests for additional information). The alternative is to make sure certs from your SP (service provider) in our case the CCM server are trusted by Shibboleth.
   a. As root copy the Shibboleth-AnyCert.jar file, which is a security provider for java, to your JRE lib/ext directory.
   ```
   wget http://www.switch.ch/aai/downloads/Shibboleth-AnyCert.jar
   cp Shibboleth-AnyCert.jar ${JAVA_HOME}/jre/lib/ext/Shibboleth-AnyCert.jar
   ```
   b. Add the following security provider to ${JAVA_HOME}/jre/lib/security/java.security by adding a line to the security.provider section:
   ```
   security.provider.7=edu.internet2.middleware.shibboleth.quickInstallIdp.
   AnyCertProvider
   ```

   > ⓘ Step c. is included in Step 8 below when you configure the whole connector. If you do not have an existing truststore, you must create a truststore first.

    c. In your ${CATALINA_HOME}/conf/server.xml file, configure the Attribute Authority Connector that needs client authentication to use the AnyCert truststore algorithm as displayed in the following example:

```
<Connector
    truststoreFile="/etc/shibboleth/truststore.jks"
    truststorePass="$TRUSTSTORE_PASSWORD$"
    truststoreAlgorithm="AnyCert"
/>
```

    Although Tomcat should never use the truststore for this connector, it is also essential to specify a truststore that contains at least one certificate, even if it is a dummy certificate.

8. Add connectors to Tomcat server.xml.
    a. **sudo vi /usr/local/tomcat6/conf/server.xml**

    b. Add the connector as below. Replace "PASSWORD" with the password you entered for the IDP key during installation. If you install shibboleth to a different location make sure you update the path in red.

```
<Connector port="443"
           protocol="org.apache.coyote.http11.Http11Protocol"
           SSLImplementation="edu.internet2.middleware.security.tomcat6.
             DelegateToApplicationJSSEImplementation"
           scheme="https"
           SSLEnabled="true"
           clientAuth="want"
           keystoreFile="/opt/shibboleth-idp/credentials/idp.jks"
           keystorePass="PASSWORD"
 truststoreFile="/opt/shibboleth-idp/credentials/trustore.jks"
 truststorePass="osmosix"
 truststoreAlgorithm="AnyCert"
/>
```

9. Instead of copying the WAR file to the ...webapps/ROOT/ location which will expand the WAR file Shibboleth recommends using a context deployment fragment.
    a. Create the file TOMCAT_HOME/conf/Catalina/localhost/idp.xml "sudo vi /usr/local/tomcat6/conf/Catalina/localhost/idp,xml

    b. Add the context information below. If you install to a different location update the path in red.

```
<Context docBase="/opt/shibboleth-idp/war/idp.war"
         privileged="true"
         antiResourceLocking="false"
         antiJARLocking="false"
         unpackWAR="false"
         swallowOutput="true" />
```

Configure Shibboleth

There are basically four configuration files for Shibboleth. All reside in /opt/shibboleth-idp/conf/. See IdPAuthUserPass for additional information.

- attribute-resolver.xml is where we define our authentication sources (in this case Active Directory) and what attributes to pull for users.
- attribute-filter.xml is where we define what user attributes to release to what SPs.
- handler.xml we define how we are going to handle user authentication/sessions.
- login.config is where we define how to authenticate users.

1. cd /opt/shibboleth-idp/conf
2. Add the following to login.config. Update items in red, so they are correct for your active directory setup:
    a. Host is the Active Directory Global Catalog. You can enter multiple servers separated with a space to provide failover/redundancy. If you use multiple hosts for authentication use need to use the same set of multiple servers for attribute resolving
    b. Base is your search base. Update to reflect your domain name. If all users reside under the default Users folder in AD you could use cn=Users,dc=cliqrtech,dc=local. If user accounts reside in different areas of the domain you may need to use the root or the directory.
    c. Contact CliQr Support for the username\password for this account.
    d. If you need to authenticate against multiple LDAP (AD) directories or disjunctive search bases in the same directory, configure you login.config. See IdPAuthUserPass for additional information.
    e. A sample login page is located, in the IdP distribution, at *src/main/webapp/login.jsp*. For information on how to customize the login page, see IdPAuthUserPassLoginPage.

```
ShibUserPassAuth {
edu.vt.middleware.ldap.jaas.LdapLoginModule required
host="ad01.cliqr.com"
port="3268"
base="dc=cliqrtech,dc=local"
ssl="false"
userFilter="sAMAccountName={0}"
serviceUser="saml@cliqrtech.local"
serviceCredential="CliQrWin2day!"
subtreeSearch="true"
referral=”follow”;
};
```

3. Modify handler.xml
   a. Find and enable the UsernamePassword LoginHandler. Remove the <!-- before the section and the --> after the section to enable the LoginHandler.

```
<LoginHandler xsi:type="UsernamePassword"
   jaasConfigurationLocation="file:///opt/shibboleth-idp/conf/login.config">
<AuthenticationMethod>rn:oasis:names:tc:SAML:2.0:ac:classes:
PasswordProtectedTransport</AuthenticationMethod>
</LoginHandler>
```

   b. Find and disable the RemoteUser Login Handler. To comment out put <!-- in front and -->at the end of the following section:

```
<LoginHandler xsi:type="RemoteUser"
</LoginHandler>
```

4. Edit attribute-resolver.xml
   a. Add an LDAP Data Connector to point to Active Directory used to resolve users and their attributes. Enter the following information into the file below
      <!-- LDAP Connector -->.
      The DataConnecter ID your choice, but must be unique to ensure user authentication against multiple sources so you can add more Connectors. You must also reference the ID when you add attributes to resolve users.

```
<resolver:DataConnector id="cliqrLDAP" xsi:type="LDAPDirectory" xmlns="urn:mace:shibboleth:2.0:resolver:dc"
ldapURL="ldap://10.100.1.220:3268" baseDN="dc=cliqrtech,dc=local"
principal="saml@cliqrtech.local"
principalCredential="CliQrWin2day!">
<FilterTemplate>
<![CDATA[
(sAMAccountName=$requestContext.principalName)
]]>
</FilterTemplate>
<LDAPProperty name="java.naming.referral" value="follow"/>

</resolver:DataConnector>
```

   b. See ResolverLDAPDataConnector for advance configuration information (using redundant LDAP servers for a connector, filtering, and so forth).
   c. Add the Name Identifier attribute. The file has an attribute already set to use the transientId. You must set a persistent ID (mapping to the sAMAccountName) as CliQr maps the external ID to this attribute (may cause problems if this ID is transient).
   d. Ensure to reference the connector you created under Dependency ref:

```
 <resolver:AttributeDefinition id="sAMAccountName" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
            sourceAttributeID="sAMAccountName">

       <resolver:Dependency ref="cliqrLDAP" />

       <resolver:AttributeEncoder xsi:type="enc:SAML1StringNameIdentifier" nameFormat="urn:mace:shibboleth:1.0:nameIden
tifier"/>

       <resolver:AttributeEncoder xsi:type="enc:SAML2StringNameID" nameFormat="urn:oasis:names:tc:SAML:2.0:nameid-for
mat:persistent"/>

   </resolver:AttributeDefinition>
```

   e. Add Attribute Definition resolvers for all applicable user attributes:
      - CliQr requires the four attributes: the user's first name, last name, email, and User ID (UID).

- If you want to setup SSO at the root tenant-level and have it also work for first-level sub-tenants, you need an attribute to mark the tenant to which a user should belong.
- If you want SSO to work for the second-level sub-tenants, you need one more attribute.

f. Add this under the Data Connector you created to pull the four required attributes.

> Reference the connector created under Dependency ref=. If you use multiple connectors, you will have multiple copies of the attribute resolvers changing the Dependency ref.

```
<resolver:AttributeDefinition id="mail" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
sourceAttributeID="mail">
<resolver:Dependency ref="cliqrLDAP" />
<resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:mace:dir:attribute-def:mail" />
<resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="mail" />
</resolver:AttributeDefinition>

<resolver:AttributeDefinition id="givenName" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
sourceAttributeID="givenName">
<resolver:Dependency ref="cliqrLDAP" />
<resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:mace:dir:attribute-def:givenName" />
<resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="givenName" />
</resolver:AttributeDefinition>

<resolver:AttributeDefinition id="sn" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
sourceAttributeID="sn">
<resolver:Dependency ref="cliqrLDAP" />
<resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:mace:dir:attribute-def:sn" />
<resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="sn" />
</resolver:AttributeDefinition>

<resolver:AttributeDefinition id="uid" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
sourceAttributeID="uid">
<resolver:Dependency ref="cliqrLDAP" />
<resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:mace:dir:attribute-def:uid" />
<resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="uid" />
</resolver:AttributeDefinition>
```

g. All the required attributes (at a minumum, the four CliQr-required attributes) are published to the Global Catalog. Not all attributes are published automatically. If you use additional attributes, make sure that these attributes are also published to the Global Catalog.

h. See the default attributes inside AD and include a Global Catalog column, so you can verify if it in the Global Catalog by default.

i. To change attribute replicate, see Global Catalogs and the Partial Attribute Set.

j. Add a new schema class or attribute definition.

k. If you want to configure SSO to authenticate users from multiple AD domains, see IdPMultipleLDAP.

5. Edit attribute-filter.xml:

   a. Add the following AttributeRules to release the attributes we are going to resolve to the SP (in this case, CliQr). Add these after the AttributeFilterPolicy to Release the transient ID to anyone

   b. If you pulled any other attributes add a rule to release them as well.

```
<afp:AttributeRule attributeID="sAMAccountName">
<afp:PermitValueRule xsi:type="basic:ANY"/>
</afp:AttributeRule>

<afp:AttributeRule attributeID="mail">
<afp:PermitValueRule xsi:type="basic:ANY"/>
</afp:AttributeRule>

 <afp:AttributeRule attributeID="sn">
  <afp:PermitValueRule xsi:type="basic:ANY"/>
 </afp:AttributeRule>

 <afp:AttributeRule attributeID="givenName">
<afp:PermitValueRule xsi:type="basic:ANY"/>
</afp:AttributeRule>

 <afp:AttributeRule attributeID="description">
 <afp:PermitValueRule xsi:type="basic:ANY"/>
</afp:AttributeRule>
```

6. Restart Tomcat :
   cd /usr/local/tomcat6/bin ./shutdown.sh
   ./startup.sh

Use Case: ADFS SAML SSO

- Overview
- Domain and Portal Verification
- CliQr Support
- CliQr SAML Authentication Configuration
- ADFS Trust Settings

Overview

> **CloudCenter does not authenticate directly to LDAP or AD.**
>
> CloudCenter only interacts with LDAP/AD through a SSO IDentity Provider (IDP) that supports SAML 2.0 protocol (for example, Ping Identity, ADFS, Shibboleth, and so forth).
>
> To implement SSO using CloudCenter:
>
> 1. You must then configure the CCM to re-direct the authentication to the SSO IDP.
> 2. You must also map some additional user custom properties (returned by the SAML IDP) to the user activation profile.
> 3. Once you complete all these steps successfully, CloudCenter automatically assigns the proper user group membership and additional roles and permissions.

A CCM instance supports Security Assertion Markup Language (SAML) 2.0 SSO through Spring Security SAML Extension.

The SysAdmin can be set up SAML integration at the root level or the tenant level. To accurately configure this integration, you must have the following information for the root tenant or sub-tenant (as applicable to your deployment):

- Tenant Information
- Activation Profile information

Domain and Portal Verification

Verify and ensure that the following information is accurate:

- The timezone and time of the CCM (and by association all other appliances) matches the AD Domain Controllers.
- The logon for the FQDN portal page (for example, https://cloud.core.enterpise.com) is accurate.

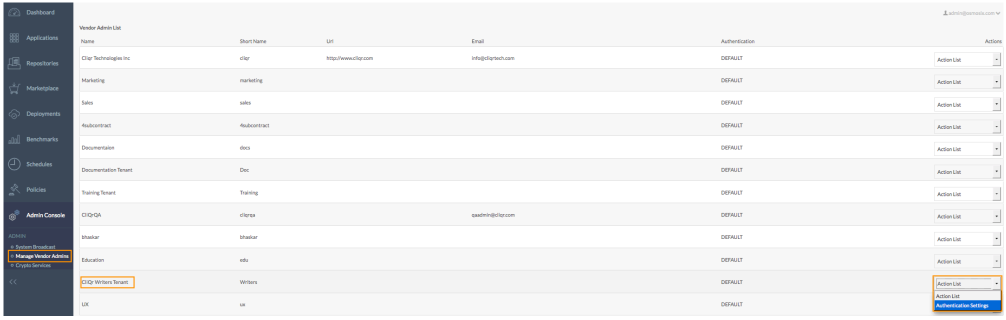
CliQr Support

Contact CliQr Support for additional information.

CliQr SAML Authentication Configuration

To configure a tenant to use SSO, follow this procedure:

1. Create a tenant (see Add Sub-Tenants)
    a. Short Name – give a string without white spaces and special characters.
    b. External Id – enter the ID of the organization in the external system with which the CliQr tenant is associated.
    c. Tenant – the CCM server domain name alias for the tenant. This will serve as the end point of the Service Provider (SP) from the SSO perspective. See Add Sub-Tenants for additional context.
2. Login as the newly created tenant admin and create an Activation Profile (see Activation Profiles).
3. Click the **Vendor Info** tab and select the newly created activation profile as Default Activation Profile.
4. Login as SysAdmin, click the **Manage Vendor Admins** tab and select the **Authentication Settings** action dropdown for this tenant.



Vendor Admin List

Name	Short Name	Url	Email	Authentication	Actions
Cliqr Technologies Inc	cliqr	http://www.cliqr.com	info@cliqrtech.com	DEFAULT	Action List
Marketing	marketing			DEFAULT	Action List
Sales	sales			DEFAULT	Action List
4subcontract	4subcontract			DEFAULT	Action List
Documentaion	docs			DEFAULT	Action List
Documentation Tenant	Doc			DEFAULT	Action List
Training Tenant	Training			DEFAULT	Action List
CliQrQA	cliqrqa		qaadmin@cliqr.com	DEFAULT	Action List
bhaskar	bhaskar			DEFAULT	Action List
Education	edu			DEFAULT	Action List
CliQr Writers Tenant	Writers			DEFAULT	Action List
UX	ux			DEFAULT	

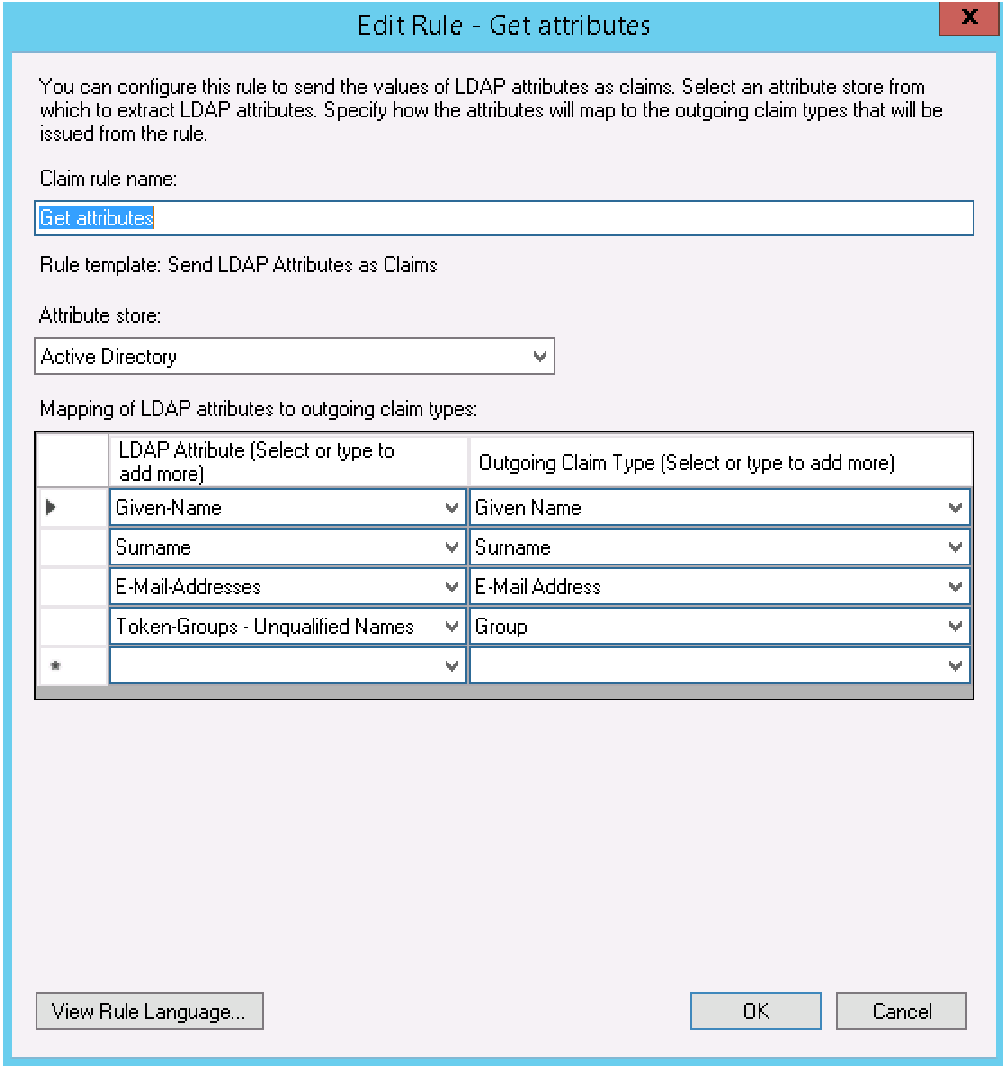
5. Enter the information in the IDP Settings:
    a. **IDP Name** (sample name is indicative of supporting AD domain)
    b. **IDP Metadata URL** – to establish the mutual trust between CliQr and the IDP (currently, this does not support HTTPS, so use HTTP).
    c. **IDP Metadata File** (if applicable)
6. Enter the information in the SP Settings:
    a. **Entity ID** – the target domain name for this authentication (should be DNS name of logon page)
    b. **Default SSO** Binding should be left at post
    c. **Logout Target URL** – If logging into your company's SAML page, you must specify the URL of the page that you want the logged in users to be redirected to when they log out of the SAML page (could be same as Entity ID)
7. Enter the information in the Attribute Mappings sections – These are the fields from the IDP that will be mapped to user attributes within CliQr. If you are unsure about these fields, please contact your IDP administrator. At a minimum, you need to provide the first name, last name, email address, and external User ID.
    a. Enter the **First Name** Mapping (http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname)
    b. Enter the **Last Name** Mapping (http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname)
    c. Enter the **Email** Mapping (http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress)
    d. Enter the **User Group** Mapping (http://schemas.xmlsoap.org/claims/Group)
    e. Download the Metadata file.
8. Click **Submit**.

ADFS Trust Settings

To configure the ADFS trust settings and to edit the corresponding claim rules, follow this procedure.

1. In the AD FS Manager, under **AD FS > Trust Relationships** > **Relying Party Trusts**, click **Add Relying Party Trust** to open the Add Relying Party Trust Wizard.
2. On the **Welcome** page, click **Start**.
3. On the **Select Import Data from a file** page, browse for and select the sp-xxxxx.xml file.
4. Click **Next**.
5. Provide a **Display name**.
6. Click **Next**.
7. On the **Configure Multi-factor Authentication Now?** page, select **I do not want to configure multi-factor authentication settings for this relying party trust at this time.**
8. Click **Next**.
9. On the **Choose Issuance Authorization Rules** page, select either **Permit all users to access this relying party**.
10. Click **Next**.
11. On the **Ready to Add Trust** page, enter the properties of the new Relaying Party Trust and click **Next** to save your relying party trust information.
12. On the **Finish** page, click **Close**. This action automatically displays the **Edit Claim Rules** box.
13. Click **Properties**.
14. On the **Advanced** tab, in the **Secure hash algorithm** list, select **SHA-1**, and then click **OK**.
15. Click the trust in the list where you want to create a claim rule.
16. Right-click the selected trust, and then click **Edit Claim Rules**.
17. On the **Select Rule Template** page, under **Claim rule template**, select **Send LDAP Attributes as Claims** from the list, and then click **Next**.
18. On the **Configure Rule** page under **Claim rule name** type ***Get Attributes*** in the display name field.
19. Under the **Mapping of LDAP attributes to outgoing claim types** select the following **LDAP Attribute** and corresponding **Outgoing Claim Type** types from the drop-down lists.

a. Given-Name = **Given Name**
b. Surname = **Surname**
c. E-Mail-Addresses = **E-Mail Address**
d. Token-Groups - Unqualified Names = **Group**



Edit Rule - Get attributes

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

Claim rule name:

Get attributes

Rule template: Send LDAP Attributes as Claims

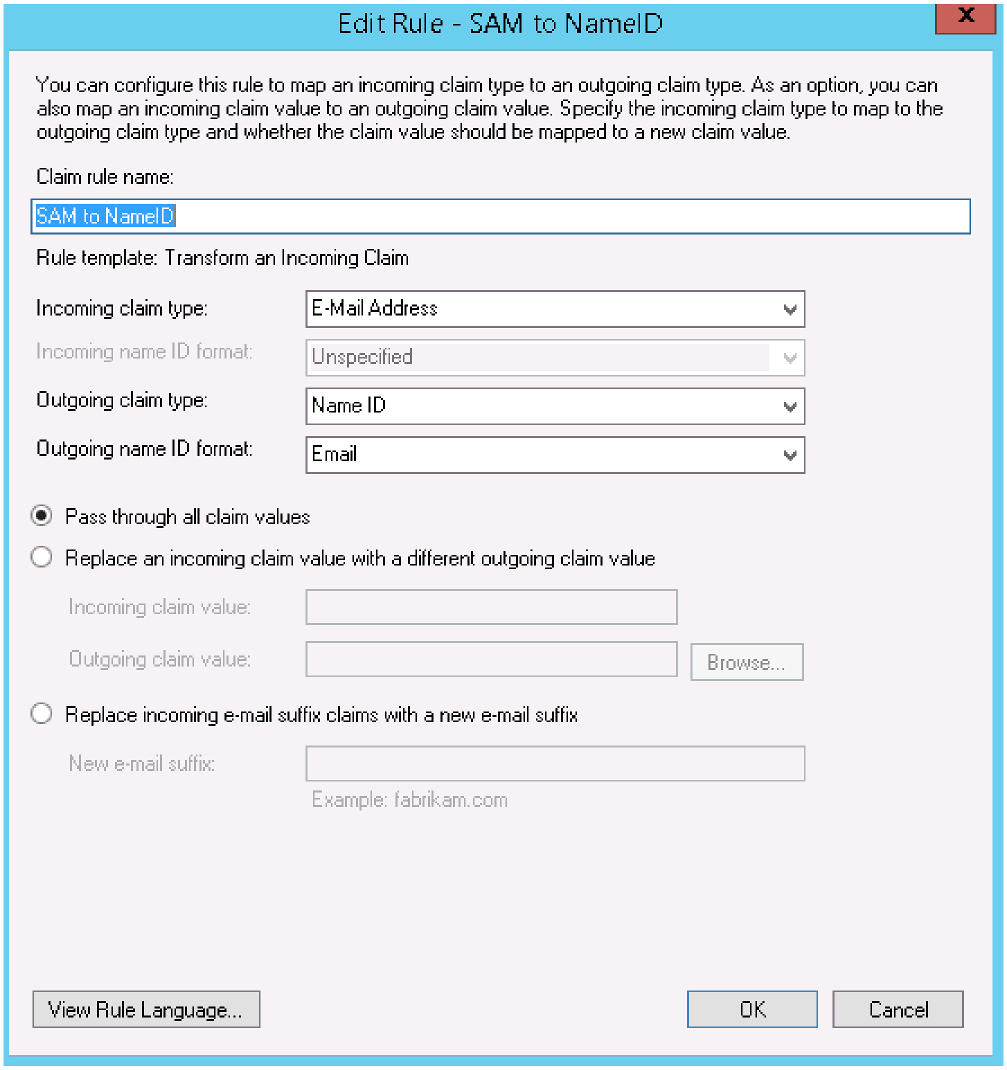
Attribute store:

Active Directory

Mapping of LDAP attributes to outgoing claim types:

	LDAP Attribute (Select or type to add more)	Outgoing Claim Type (Select or type to add more)
▶	Given-Name	Given Name
	Surname	Surname
	E-Mail-Addresses	E-Mail Address
	Token-Groups - Unqualified Names	Group
*		

20. Click **OK**.
21. Add another rule, to the Transform an Incoming Claim template – on the **Select Rule Template** page, under **Claim rule template**, select **Transform an Incoming Claim** from the list, and then click **Next**.
22. Name the rule as SAM to NameID and map the following values:
    a. Incoming claim type = **E-Mail Address**
    b. Outgoing claim type = **Name ID**
    c. Outgoing name ID format = **Email**



You can configure this rule to map an incoming claim type to an outgoing claim type. As an option, you can also map an incoming claim value to an outgoing claim value. Specify the incoming claim type to map to the outgoing claim type and whether the claim value should be mapped to a new claim value.

Claim rule name: SAM to NameID

Rule template: Transform an Incoming Claim

Incoming claim type: E-Mail Address

Incoming name ID format: Unspecified

Outgoing claim type: Name ID

Outgoing name ID format: Email

● Pass through all claim values

○ Replace an incoming claim value with a different outgoing claim value

Incoming claim value:

Outgoing claim value:

○ Replace incoming e-mail suffix claims with a new e-mail suffix

New e-mail suffix:

Example: fabrikam.com

23. Click **OK**.

You have now configured the ADFS SAML SSO integration.