# Secure Agent Communications

- CLOUDSTACK-9993 - Getting issue details... STATUS



**Bug Reference**

**CLOUDSTACK-9993 - Getting issue details...** STATUS

## Introduction

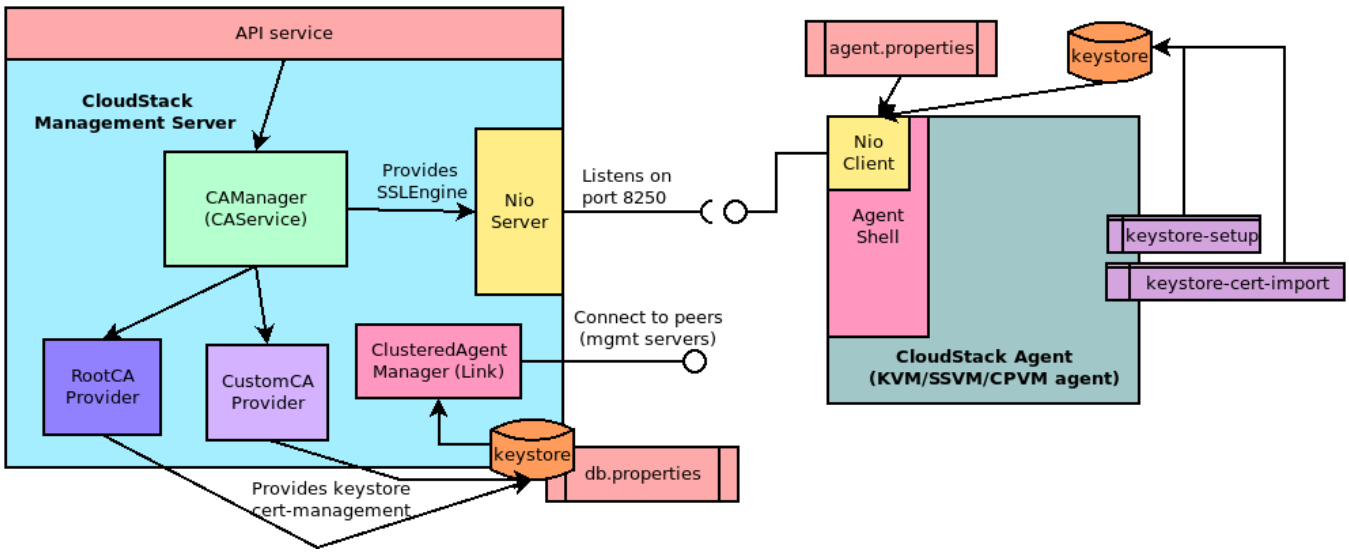
In current CloudStack, the agent-management server communication is weakly secured by one way SSL authentication while encrypted (on port 8250 /default) and allows for any client/agent to connect and be served by the management server. There are other services that need TLS/SSL security and upcoming features such as container/application service etc. require certificate management. The common issue is CloudStack has no certificate management to provide security for its internal component especially the agent-mgmt server and mgmt-mgmt server communication. The aim of this feature is to provide pluggable CA (certificate authority) management in CloudStack that can fetch/provision certificates to (new) host(s) and systemvms. As a default CA plugin, a root CA plugin will be implement where CloudStack becomes a self-signed Root Certificate Authority. Developers will have option to implement further integration with their TLS/SSL cert providers such as letsencrypt and other vendors.

## Feature Specification

• A pluggable CA service framework will be implemented that provides pluggable means to initialize SSLContexts/SSLEngines in the management server. The plugins can implement their own trust management to validate and verify incoming client request and security.
• A CA plugin can have its own specific settings, API etc.
• The pluggable CA service framework will enable a custom CA plugin to implement its own background task service to monitor active agents/client list with expiring certificates, failing authentication, send out alerts etc.
• Certificate lifecycle operations such as to create, renew, revoke, provision/propagate certificates will be provided by the framework, however the mechanisms will be implemented by a CA plugin.
• A root CA plugin, 'root', will be based on a self-signed root certificate that is created by the management server itself during initialisation. This plugin will have configurations and certificate data available in the database and provide means to create, renew, provision/propagate certificates to clients/agents. The root CA plugin will maintain an internal certificate revocation list, and provide means for automatic renewal and propagation of certificates to valid /active agents. Parts of expiring clients detection and renewal using a background thread may be provided by the framework.

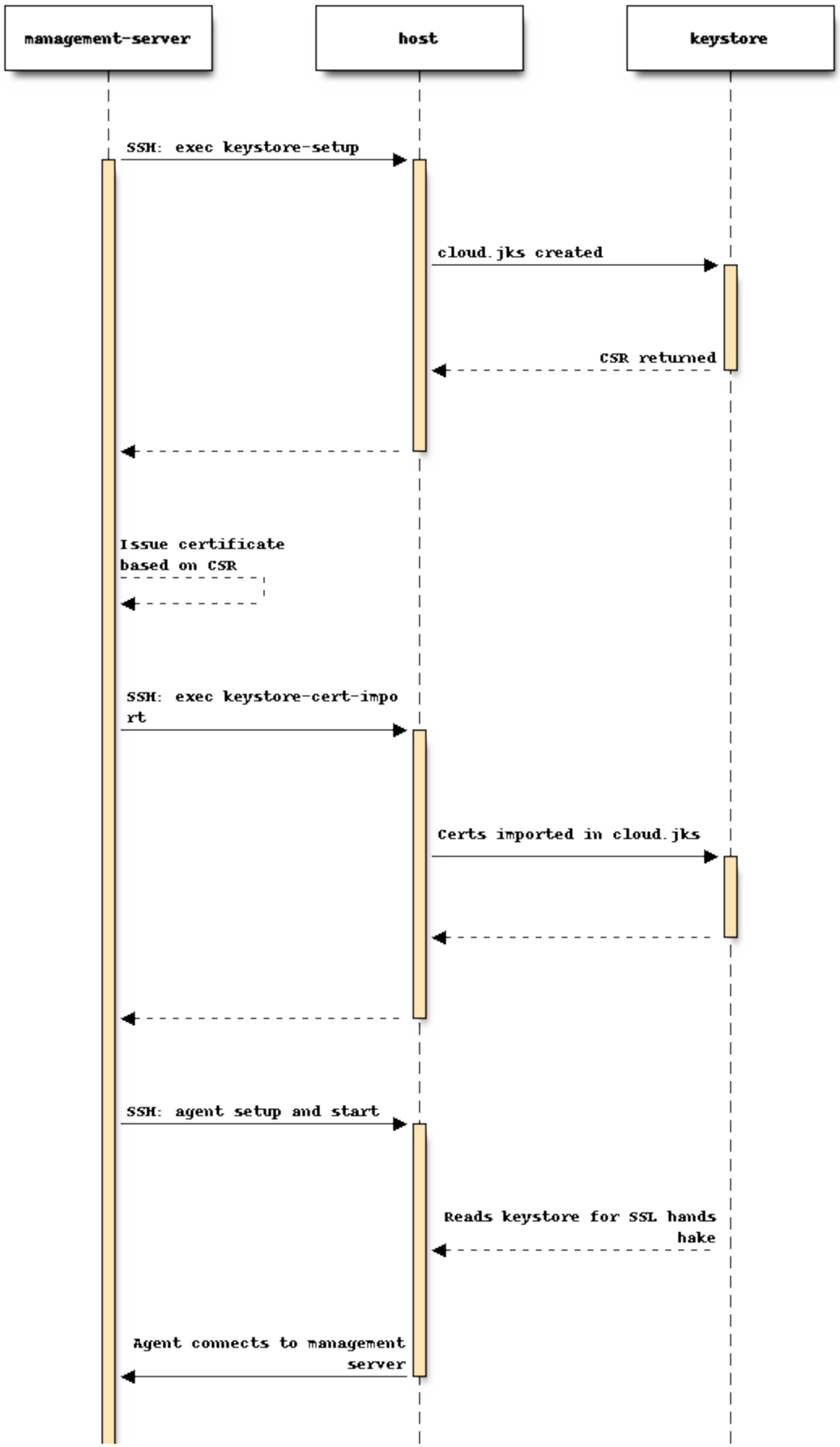
• Server-side components that use the NioServer class, on initialization can have a keystore (a .keystore file, db-backed or externally) initialized by the configured CA plugin along with custom trust managers The root CA plugin will use db/file backed keystore, while the default CA plugin will continue to use a file-based keystore. Other implementations are out of scope of this FR. This also means that the root CA plugin will only be enabled for servers running in the management server(s) and not in agents on other systems.
• Client-side components that use the NioClient class such as the (CPVM/SSVM/KVM) agents will have access to a keystore file, such as the config location /etc/cloudstack/agent. The keystore file can be used by in-band (agent) and out-of-band mechanisms to add/remove/update client certificates.
• The CA service framework will add hooks to the resource manager to carry out host-trust initialization (when hosts are added) if the configured CA plugin supports such operations. The CA service, along with the configured CA plugin can use ssh and other means to initialize client certificates on a (KVM) host such as to provide means of capturing generated CSRs and creating/propagating the client certificates to the agent (host). A new command-answer pattern will be introduced for propagation and on- boarding of certificate with already connected/active agents.
• A new script will be implemented to allow for generation of client-side certificate and CSR. The script can be executed by in-band or out-of-band mechanisms, and participate in certificate provisioning operations.
• A (comma-separated) list of management servers can be supported in agents, where on connection/disconnection/reconnection they can cycle through the list. This may remove dependency of an explicit LB for port 8205 over multiple-management servers. However, the configuration of existing agents by the management server(s) with such a list of addresses of management servers, is out of scope of this FR. It may be discussed and addressed in a separate FR.
• The enforcement of the certificate requirement will be optional. The CA plugin can provide a strictness setting for the enforcement of client-side certificate validation. This will allow for a strategy to on-board an existing CloudStack deployment to use a new CA plugin with client-authentication and validation disabled until all hosts are updated and configured to use the CA plugin-provided certificates (provisioned in-band or out-of-band way).

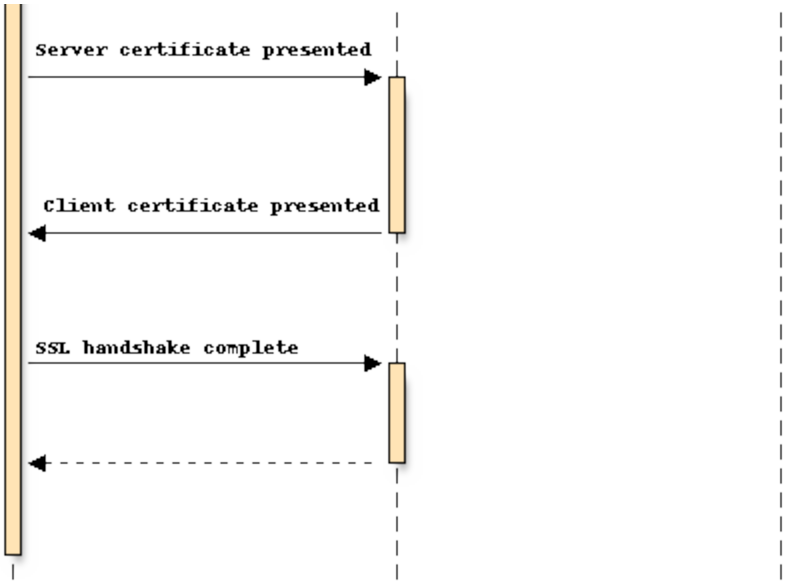
## High level component diagram



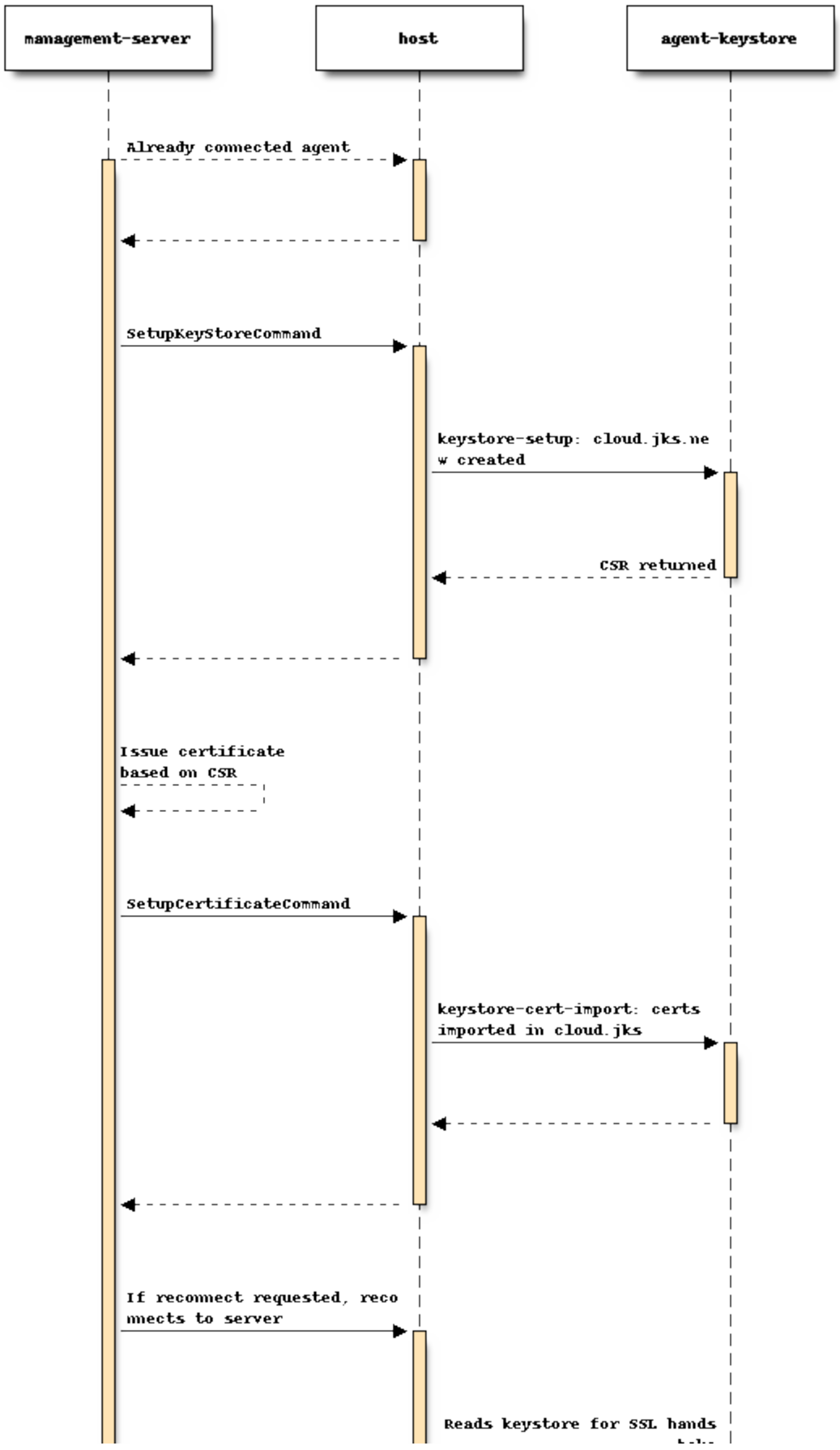

## Sequence Diagrams

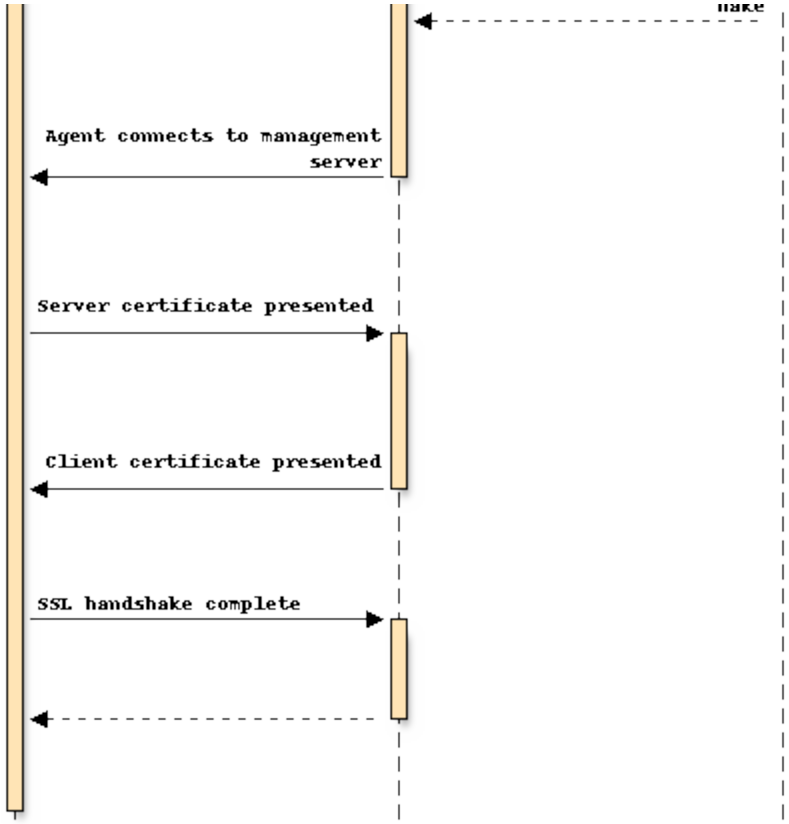
**S1. SSH-based provisioning on the addition of (KVM) hosts and system VMs:**


management-server
host
keystore
SSH: exec keystore-setup
cloud.jks created
CSR returned
Issue certificate based on CSR
SSH: exec keystore-cert-import
Certs imported in cloud.jks
SSH: agent setup and start
Reads keystore for SSL handshake
Agent connects to management server

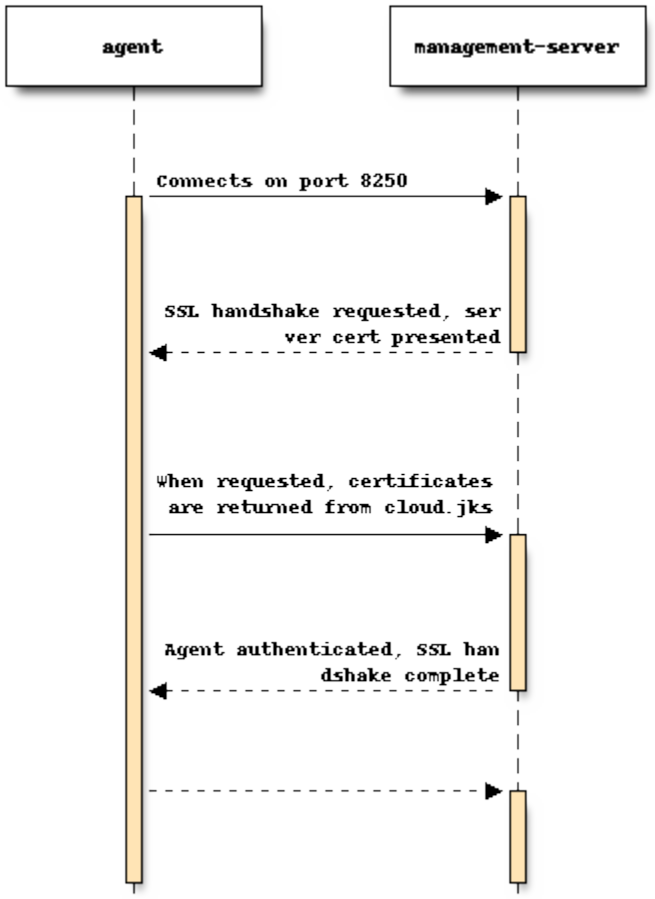



**S2. Provisioning on existing/connected agents:**


management-server
host
agent-keystore
Already connected agent
SetupKeyStoreCommand
keystore-setup: cloud.jks.new created
CSR returned
Issue certificate based on CSR
SetupCertificateCommand
keystore-cert-import: certs imported in cloud.jks
If reconnect requested, reconnects to server
Reads keystore for SSL hands




**S3. SSL handshake and negotiations on connection/re-connection:**




## APIs

The framework will implement following APIs that accept a plugin option, these APIs provide certificate lifecycle management operation as supported by the plugin and may throw a not-supported exception: (some names may be changed during implementation, we'll update the FS accordingly)

o listCAProviders: returns the list of available CA provider plugins.
o listCaCertificate: (plugin specific) returns concatenated (single/chained) root public certificate that can be imported by clients, browser etc.
o issueCertificate: (plugin specific) creates and returns a client certificate.
o revokeCertificate: (plugin specific) revokes a previously created client certificate.
o provisionCertificate: (plugin specific) renews an expiring/expired client certificate.

## UI changes

- Button to copy/download root public certificate as supported by the configured CA plugin.
- Events, alerts in the UI.

## Global Settings

Global settings for the CA framework:

- ca.framework.provider.plugin: The configured CA provider plugin
- ca.framework.cert.keysize: The key size for certificate generation
- ca.framework.cert.signature.algorithm: The certificate signature algorithm
- ca.framework.cert.validity.period: Certificate validity in days
- ca.framework.cert.automatic.renewal: Certificate auto-renewal setting

- ca.framework.background.task.delay: CA background task delay/interval
- ca.framework.cert.expiry.alert.period: Days to check and alert expiring certificates

Global settings for the default 'root' CA provider:

- ca.plugin.root.private.key: (hidden/encrypted) CA private key
- ca.plugin.root.public.key: (hidden/encrypted) CA public key
- ca.plugin.root.ca.certificate: (hidden/encrypted) CA certificate
- ca.plugin.root.issuer.dn: The CA issue distinguished name
- ca.plugin.root.auth.strictness: Are clients required to present certificates
- ca.plugin.root.allow.expired.cert: Are clients with expired certificates allowed

### Commentary

- Comma separate list of management server IPs can be set to the 'host' global setting. Newly provisioned agents (KVM/CPVM/SSVM etc) will get radomized comma separated list to which they will attempt connection or reconnection in provided order. This removes need of a TCP LB on port 8250 (default) of the management server(s).
- All fresh deployment will enforce two-way SSL authentication where connecting agents will be required to present certificates issued by the 'root' CA plugin.
- Existing environment on upgrade will continue to use one-way SSL authentication and connecting agents will not be required to present certificates.
- A script `keystore-setup` is responsible for initial keystore setup and CSR generation on the agent/hosts.
- A script `keystore-cert-import` is responsible for import provided certificate payload to the java keystore file.
- Agent security (keystore, certificates etc) are setup initially using SSH, and later provisioning is handled via an existing agent connection using command-answers. The supported clients and agents are limited to CPVM, SSVM, and KVM agents, and clustered management server (peering).
- Certificate revocation does not revoke an existing agent-mgmt server connection, however rejects a revoked certificate used during SSL handshake.
- Older `cloudstackmanagement.keystore` is deprecated and will no longer be used by mgmt server(s) for SSL negotiations and handshake. New keystores will be named `cloud.jks`, any additional SSL certificates should not be imported in it for use with tomcat etc. The `cloud.jks` keystore is stricly used for agent-server communications.
- Management server keystore are validated and renewed on start up only, the validity of them are same as the CA certificates.

## DIY CA Provider Plugin

The following interface needs to be implemented by a new CA provider plugin:

https://github.com/apache/cloudstack/blob/master/framework/ca/src/org/apache/cloudstack/framework/ca/CAProvider.java

See the default CA (root) provider plugin as an example on how to write a CA provider plugin:

https://github.com/apache/cloudstack/blob/master/plugins/ca/root-ca/src/org/apache/cloudstack/ca/provider/RootCAProvider.java