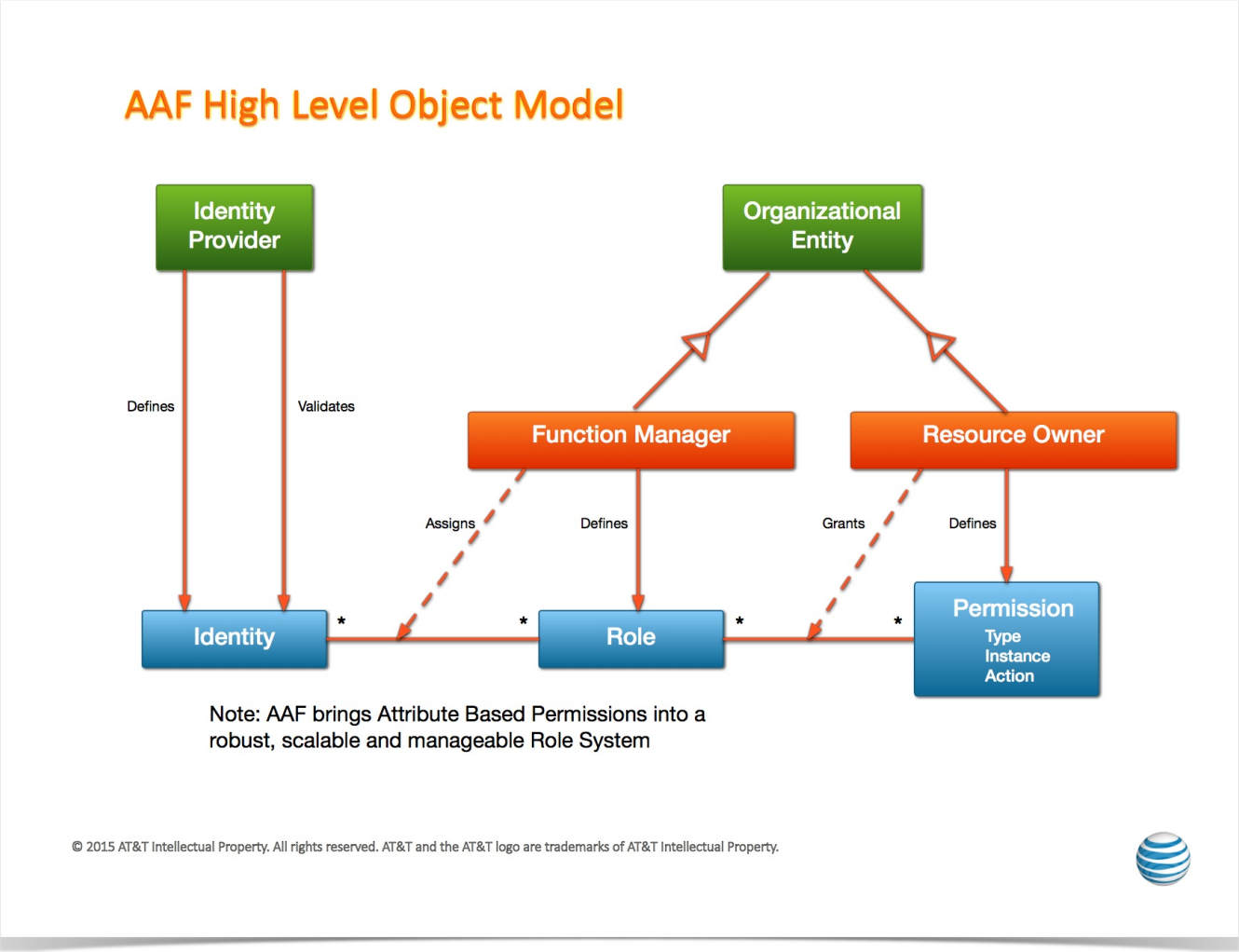
Application Authorization Framework (AAF) Documentation

https://bestpractices.coreinfrastructure.org/projects/2303/badge

AAF is designed to cover Fine-Grained Authorization, meaning that the Authorizations provided are able to used an Application's detailed authorizations, such as whether a user may be on a particular page, or has access to a particular Pub-SUB topic controlled within the App.

This is a critical function for Cloud environments, as Services need to be able to be installed and running in a very short time, and should not be encumbered with local configurations of Users, Permissions and Passwords.

To be effective during a computer transaction, Security must not only be secure, but very fast. Given that each transaction must be checked and validated for Authorization and Authentication, it is critical that all elements on this path perform optimally.



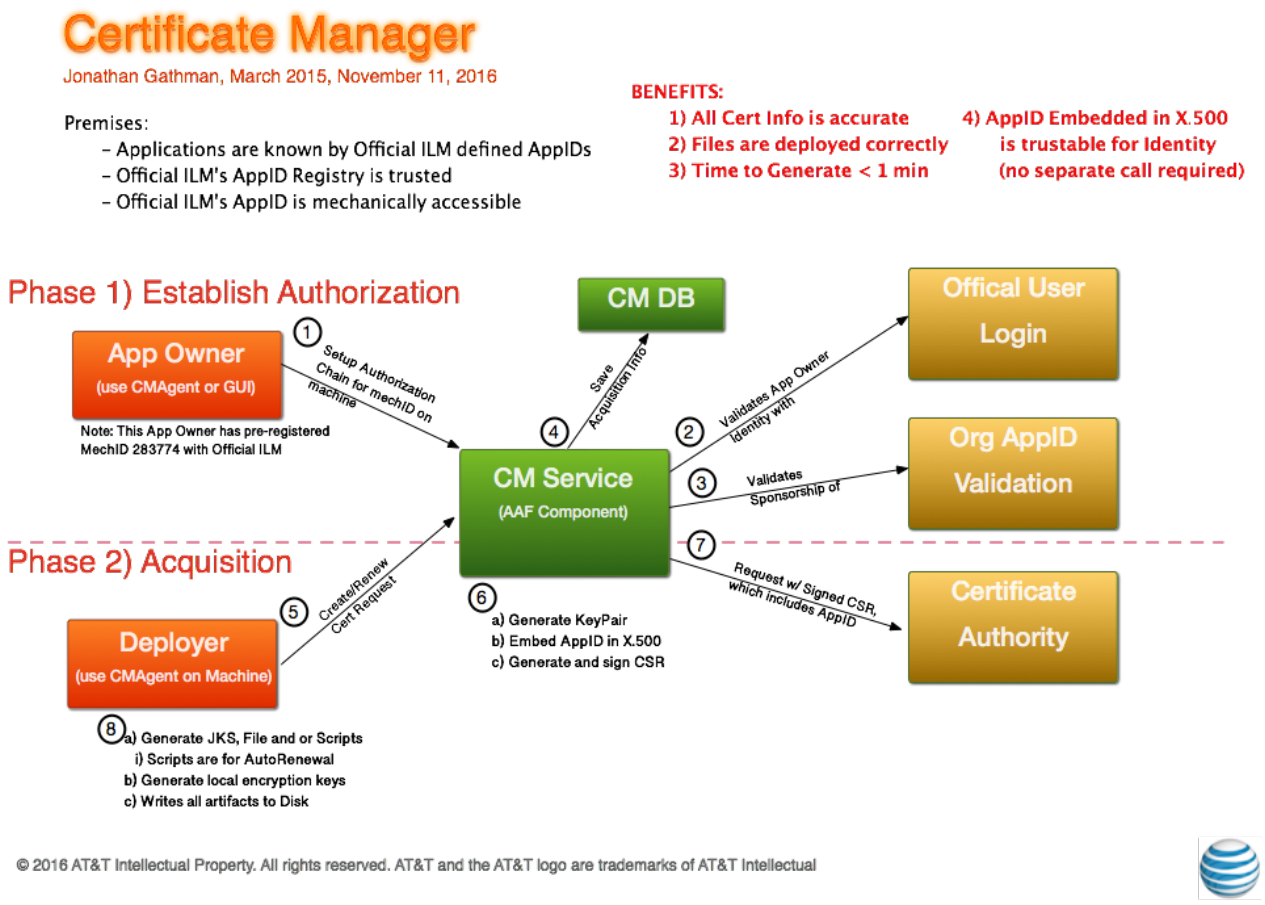
Certificate Manager

Overview

Every secure transaction requires 1) Encryption 2) Authentication 3) Authorization.

- HTTP/S provides the core Encryption whenever used, so all of AAF Components require HTTP/S to the current protocol standards (current is TLS 1.1+ as of Nov 2016)
    - HTTP/S requires X.509 certificates at least on the Server at minimum. (in this mode, 1 way, a client Certificate is generated)
    - Certificate Manager can generate certificates signed by the AT&T Internal Certificate Authority, which is secure and cost effective if external access are not needed
    - These same certificates can be used for identifying the Application during the HTTP/S transaction, making a separate UserID/Password unnecessary for Authentication.
- Authentication - In order to tie generated certificates to a specific Application Identity, AAF Certificate Manager embeds an Identity Lifecycle Management (ILM) AppID in the Subject. These are created by AT&T specific Internal Certificate Authority, which only generates certificates for AAF Certman. Since AAF Certman validates the Sponsorship of the AppID with requests (automatically), the end user can depend on the AppID embedded in the Subject to be valid without resorting to external calls or passwords.
    - ex:
- Authorization - AAF Certman utilizes AAF's Fine-grained authorizations to ensure that only the right entities perform functions, thus ensuring the integrity of the entire Certificate Process

Design and Mechanisms



Use CMAgent for Artifact Management

Preparation

Use latest Code Access Data Identity (CADI) Jar File - Get it from Maven Central (Open Sourced Version)

Creation

IMPORTANT! if you are using Self-Renewal processes, you MUST include "jks" for Types

CMAgent Creation

```
Create Certman Artifact# FOR TEST
```

```
java -jar /opt/app/cadi/1.3.1/lib/cadi-aaf-1.3.2-full.jar cm_url=https://aaftest.onap.org:8150 create


UserID (dgl@openecomp.org):

Global Login Password:

AppID: a123@myapp.onap.org

Machine: xyz.com

Types (file,jks): jks

Sponsor dgl@openecomp.org

ConfigFile RootName: org.onap.myapp

CA: aaf

Directory: /opt/app/myapp

OS User (zz9999): zz9999

2016-07-28T09:37:48.402-0500: X509 validation turned off

2016-07-28T09:37:48.455-0500: Call to AAF Certman successful a123@myapp.onap.org, xyz.com
```

Creating & Deploying Certificates With Certman

Step 1: Prerequisites

The majority of the setup is for establishing the Application's Identity in AAF and ILM. This is required to ensure the chain of responsibility from the Certificates to the Sponsor of the AppID. If your app already uses AAF, that can be skipped. If a AppID is already established for AppID/Password, that one should be used. Do not obtain another one.

1. ILM enrolled AppID, because these are about Applications
2. AAF Namespace, so we can ensure only the right people may generate a certificate purporting to be that identity

   Steps 1 and 2 are accomplished by following these instructions: OnBoarding <Link Dead>
3. Install CADI (Latest Version) on boxes where you will use "CMAgent"
   a. Java, should be 1.8+ (1.7 still works)
   b. Direct Jar Method - this is the best way to use Certificate Manager Agent...

Step 2: Create Artifacts

The App Owner (Should be the Namespace Owner AND the Sponsor of Record of the AppID in ILM Records). Follow these instructions: GUI Instructions

NOTE:

- if you want self-renewing certificates, make sure to chose "jks" as an artifact type in step 7
- in step 7, "Directory" must be writable by the user listed in "O/S User". If an application is going to use cadi + certificates, ensure that the process is run by the same user as "O/S User". Information on Unix/Linux file permissions can be found by googling, for instance this post.
- in step 11 ("Copy Artifact"), make sure the list of machines you enter does not contain whitespace
- Alternatively, the GUI can be skipped by following these instructions: Use CMAgent for Artifact Management. In most cases, users should use the GUI instructions above

Step 3: Create/Assign Permission

Use AAF's GUI > Command Prompt to create the following permission and assign it to a role. Pick the GUI for the appropriate environment from Application Authorization Framework

Create Certman Permission

Replace "<ns>" with the name of the application's namespace.

Create Permission

```
perm create com.att.<ns>.certman  aaf   request
```

Grant Certman Permission to a Role

For instance if, the deployer is an admin of the app's namespace, grant the permission created above to <ns>.admin. For information on how to create a new role, or to assign a user to a role, see Documentation for Namespace Admins. Replace "<ns>" with the name of the application's namespace. Replace "<some-role" with the name of the deployer's role.

Grant Permission to Role

```
perm grant com.att.<ns>.certman aaf request <ns>.<some-role>
```

Notes:

- For details on creating roles, adding users to roles, etc, see: Documentation for Namespace Admins
- You may use your AppID's password instead of a Deployer's name, but it MUST BE FULLY QUALIFIED AppID, i.e. a123@myapp.onap.org
- Step 4: Deploy/Install Certificates

With AAF Certman, all private keys, certificates and supporting files are delivered directly to the machine in question into the directories specified by the Artifact Creation step above. The point of "CRON" is to have the Machine itself check regularly for the Certificate's expiration, and have the machine request Renewal at the right time.

The steps for the *first deployment only* are designed to be taken by either a person, in the case of Operations Personnel, or automatically. In either case, they simply need to be able to establish their Identity to the Certman process. For a Person, the program will ask for their Global Login ID. For an automated process, (i.e. VM Creator/deployer), identity must still ultimately be established for the process. After the *firs t deployment* , renewals are done automatically given the Certificate Credentials in question.

Machine Requirements:

Java 1.7+ (must be at least JDK 1.7, because communications use TLS 1.1+ per ILM Requirement, and JDK 1.6 does not natively support.)

cron (or equivalent scheduler) Cron is ubiquitous on standard Linux VMs. Other schedulers can be used, but the user must make the modifications.

These steps are for an Operations Person to perform. As stated above, a VM Creating process could perform, but the details should be worked out with Tier 3.

- Login
- *sudo*, if necessary, to the O/S User which was stated in the "Artifact Create" statement
  Run the following program in Java (it doesn't matter which directory)

Note that the ID Used must be a DEPLOYER. This means it is either the AppID itself (if User/Password exists), or someone with the appropriate AAF Permission granted to them.

Place Certificates on Disk

```
> java -jar /opt/app/cadi/1.3.2/lib/cadi-aaf-1.4.0-full.jar place a123@myapp.onap.org mymachine.domain.onap.org



  UserID (dgl@openecomp.org):

  Global Login Password:


  2016-07-28T09:47:50.194-0500: X509 validation turned off

  Reconstitute Certificates 2.319999ms

  Reconstitute Private Key 0.114307ms
```

```
# NOTE:  If you make a mistake in the URL or your Password, run the following:

> java -jar /opt/app/cadi/1.3.2/lib/cadi-aaf-1.3.2-full.jar -logout


and try again.
```

- " a123@myapp.onap.org " is the AppID reference to the Artifact
- "mymachine.domain.att.com" is the fully qualified Machine Name (FQDN), which matches an Artifiact in Certificate Manager (see above for creation)
  - See "Templates" below for Special Cases
  - For Automation purposes, CMAgent will try the Java Default for machine if not on Command Line. Whether this returns a fully qualified name (FQDN) may depend on setup. Another option for automation (in Linux) may be replacing "my machine.domain.att.com" with "`uname -n`.domain.att.com"

Results in the directory if "type=jks" would be:

Type=jks

```
-r--------  1 abc123  staff  2074 Jul 27 17:53 myns.keyfile

-r--------  1 abc123  staff  2625 Jul 28 09:47 mynsTrust.jks

-r--------  1 abc123  staff   392 Jul 28 09:47 myns.props

-r--------  1 abc123  staff  4186 Jul 28 09:47 myns.jks

-r--------  1 abc123  staff    87 Jul 28 09:47 myns.chal
```

Results in the directory if "type=file" would be:

Type=file

```
-r--------  1 abc123  staff  2074 Jul 27 17:53 myns.keyfile

-r--------  1 abc123  staff  1713 Jul 28 09:34 myns.key

-rw-r--r--  1 abc123  staff  3899 Jul 28 09:34 myns.crt

-r--------  1 abc123  staff    87 Jul 28 09:34 myns.chal
```

Results in the directory if "type=script" would be:

Type=script

```
-rw-r--r-- 1 abc123 cssari 1454 Nov  9 14:42 org.onap.myapp.check.sh

-rw-r--r-- 1 abc123 cssari  736 Nov  9 14:42 org.onap.myapp.crontab.sh
```

Notes:

-
  - Multiple Types are separated by commas: "type=jks,file,script" to get all of them.
  - Great lengths have been taken to ensure that as few special variables are required for Deployment as possible

Self Renewal

Generating the "scripts" are the key to making Certificates Self-Renewing. Always add these scripts for Unix Based Systems. You will have to work your own solution for other O/Ss that don't have "cron".

!! If you use Self Renewal, you MUST have the "jks" files generated!!

Setting up Self-Renewal:

- After deploying certificate above, test the Certificate access with the "check.sh" script
- ALL THE SELF-RENEWAL SCRIPTS ARE "bash" Scripts. If you are not running a bash shell, use "bash" as prefix
- **Check the Certificate ID is working**

```
·         # Validate that Certificate can be used to contact Certificate Manager
```

```
·         bash> bash org.onap.myapp.check.sh; cat *STD*

·         2016-11-11T07:53:24.280-0500: cadi_keyfile points to /opt/app/myapp/org.onap.myapp.keyfile

·         2016-11-11T07:53:25.256-0500: PropertyLocator enabled with https://aafcrl.test.att.com:8150

·         2016-11-11T07:53:25.572-0500: X509Certificate for a123@myapp.onap.org on xyz.com has been checked on
2016-11-11. It expires on Tue May 09 15:41:58 EDT 2017; it will not be renewed until 2017-04-09.

·         2016-11-11T07:53:25.580-0500: Trans Info

·                  REMOTE Check Certificate 322.62177ms

·

·         # --- Note that the Check Process will create 3 output files

·         bash> ls -l

·         -rw-r----- 1 abc123 cssari    0 Nov 11 07:53 org.onap.myapp.STDERR

·         -rw-r----- 1 abc123 cssari  182 Nov 11 07:53 org.onap.myapp.msg

·         -rw-r----- 1 jabc123 cssari  491 Nov 11 07:53 org.onap.myapp.STDOUT

·

·         # -- These can be checked at any time.  If successful, STDERR size should be zero, but will populate
when there is a problem.

·         # -- The "msg" file is the result of the check.sh, showing the Check Date and the Expiration Date
```

- Once the "check.sh" is ensured to work, add the nightly Check to Cron (in Unix).

1.
    a. The crontab.sh script will overwrite it's own entry, but not others
    b. The crontab.sh script will randomize minutes and hours between 1:00 am and 3:59am, local machine time.
    c. The very first night the script invokes, and email will be sent to the Artifact Notification of success/failure
        i. After that first night, emails are only sent on Failure and Renewal
    d. IF, and only if, there is a script in the same directory, **<dir>/<ns>.restart.sh**, it will be invoked on Renewal
        i. The purpose of this mechanism is because while a Certificate may be renewed and regenerated, it is unlikely the Container/App knows about it, and probably needs bouncing before the Certificate takes effect. *Not planning for this step will give you an Outage just as if you forgot to change your Certificate manually.*
        ii. This script is NOT generated, because there are too many different ways and cases for start versus non-starting of services. The End App is responsible write any such service
        iii. IF the App intends to write such a Script, it would be VERY wise to avoid the possibility of Certificate Renewal on the same night for all instances of the APP.
        iv. Good Idea: One group uses this to create a Task in their Work Queues.
        v. **AAF is not Responsible** for the App's Script working, or for what it does. YOU MUST DO your own testing, and thought processes behind restarts. Be WISE, Be Responsible

Crontab

```
# Optional... validate what is in Crontab now

bash > crontab -l

#### Someone Else's Cron info

# 3 3 3 3 3  /bin/bash /someone's nightly process

####


bash > bash com.att.myapp.crontab.sh


bash > crontab -l

#### Someone Else's Cron info

# 3 3 3 3 3  /bin/bash /someone's nightly process

####

#### BEGIN com.att.myapp Certificate Check Script ####

57 1 * * * /bin/bash /home/abc123/myapp/org.onap.myapp.check.sh >> /home/abc123/myapp/cronlog 2>&1

#### END org.onap.myapp Certificate Check Script ####
```

Other uses:

- Once the Certificates are generated, they may be used for certain accesses to Certificate Manager Agent... but only those functions given to the App

CMAgent

```
bash> java -jar /opt/app/cadi/1.3.2/lib/cadi-aaf-1.3.2-full.jar cadi_prop_files=org.onap.myapp.props read

2016-11-11T10:26:48.761-0500: cadi_keyfile points to /home/abc123/myapp/org.onap.myapp.keyfile

2016-11-11T10:26:49.746-0500: PropertyLocator enabled with https://aafcrl.test.att.com:8150

AppID:          a123@myapp.onap.org

  Sponsor:       dgl@openecomp.org

Machine:         xyz.com

CA:              aaf

Types:           jks,script

Namespace:       org.onap.myapp

Directory:       /home/zz9999/myapp

O/S User:        zz9999

Renew Days:      30

Notification     mailto:dgl@openecomp.org

2016-11-11T10:26:50.177-0500: Trans Info

         Read Artifact 429.56503ms


# Please note that the only those entities given the "Deploy" permission may see the Passwords encrypted within
the Property Files

bash> java -jar /opt/app/cadi/1.4.0/lib/cadi-aaf-1.3.2-full.jar cadi_prop_files=org.onap.myapp.props showpass

2016-11-11T10:29:29.994-0500: cadi_keyfile points to /home/zz9999/myapp/org.onap.myapp.keyfile

2016-11-11T10:29:30.896-0500: PropertyLocator enabled with https://aafcrl.test.att.com:8150

2016-11-11T10:29:31.579-0500: SVC1403 Forbidden: a123@myapp.onap.org does not have Permission.

2016-11-11T10:29:31.580-0500: Trans Info

         REMOTE Show Password 682.36285ms


bash> java -jar /opt/app/cadi/1.3.2/lib/cadi-aaf-1.3.2-full.jar cm_url=https://aaftest.onap.org:8150 showpass
a123@myapp.onap.org

Your Identity: dgl@openecomp.org     ## Must be an entity with Deploy Permission

Password:

2016-11-11T10:34:58.296-0500: PropertyLocator enabled with https://aaftest.onap.org:8150

2016-11-11T10:34:58.613-0500: Cannot validate X509 Client Validity: No TrustStore set    ## Note: this is ok, if
you're sure you have the right Certman

cadi_truststore_password=I*AM(*A*(*GENERATED(*()TRUSTSTORE&PASSWORD

cadi_key_password=I*AM(*A*(*GENERATED(*()PASSWORD

cadi_keystore_password=I*AM(*A*(*GENERATED(*()PASSWORD

ChallengePassword=I*AM(*A*(*GENERATED(*()CHALLENGE&PASSWORD

2016-11-11T10:34:59.743-0500: Trans Info

         REMOTE Show Password 1443.2354ms
```

Special Cases - Templates

Note: ILM no longer requires special exceptions for SANs. You may add them in your Artifact at creation time.

The default case for Certificate Manager is to certificates for the specific machine with Authorization. Since it is very simple to create Certificates, and have them renewed automatically, one of the main reasons for SANs is removed (the difficulty of creating and managing certificates).

Templates are only available for VM Creating Deployment tools, where it is the only solution. You must get a Permission applied by AAF Team to make this happen:

com.att.aaf.ca|aaf|domain

There are two varieties of templates that are designed to make creating certificates in today's more dynamic environment

Vanity URL:

Many services have Vanity URLs (or Round Robin). AAF is one of these. Normally, Certificate Manager validates whether the deployment call is made from the Artifact created. With the "Vanity URL Template", when deploying a new certificate, the check is made whether the incoming request for deployment is made from on of the machines in the Vanity URL, if so, then it is accepted, and a new Artifact is created for that machine is created in the Template Image for the particular machine requesting.

How this works:

The "machine" when creating the Authorization Artifact (see above), is created with the vanity URL, i.e. aaf.onap.org.

Note: Because it assumed that the Vanity URL should be part of the "SAN" list, you need to have the approval for SAN, see above.

When deployed, the new Artifact is generated from the Template, which enables tracking of renewals etc.

When Deploying, adding the "machine" on the command line is required in the format: <vanity URL>:<real machine name>[:<additional SAN>]*

Domain:

The "Domain" is a special case, used strictly by Dynamic VM creators, and similar tools. In this case, the AppID owner specifies that his AppID may deployed on any in a specific domain, such as "*.vmgroup.onap.org". This approval requires special ILM exception as well as AAF approval, and when accepted, the permission "org.onap.aaf.ca|aaf|domain" is grant

How this works:

The "machine" when creating the Authorization Artifact (see above), is created with the domain starting with *, example: "*.vmgroup1.org.onap"

When deployed, the new Artifact is generated from the Template, which enables tracking of renewals, etc.

When Deploying, adding the "machine" on the command line is required in the format: <domain>:<real machine name>[:<additional SAN>]*

Modification to "Place" command

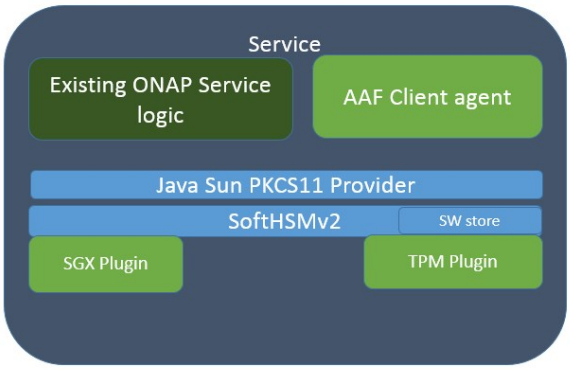
```
> java -jar /opt/app/cadi/1.3.2/lib/cadi-aaf-1.3.2-full.jar place a123@myapp.onap.org "*.vmgroup1.onap.org:
test123.vmgroup1.onap.org:san1.onap.org:san2.onap.org"
```

Hardware Security in AAF

Hardware Security in AAF

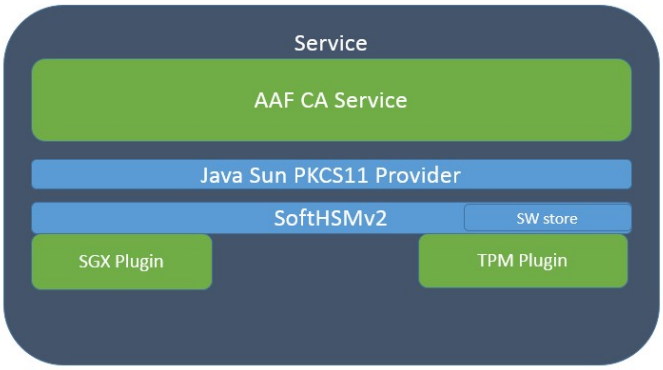
TPM/SGX Plugin Client

- TPM/SGX Client will be a plugin that will integrate with the AAF client library to provide hardware security.



TPM/SGX Plugin on Server

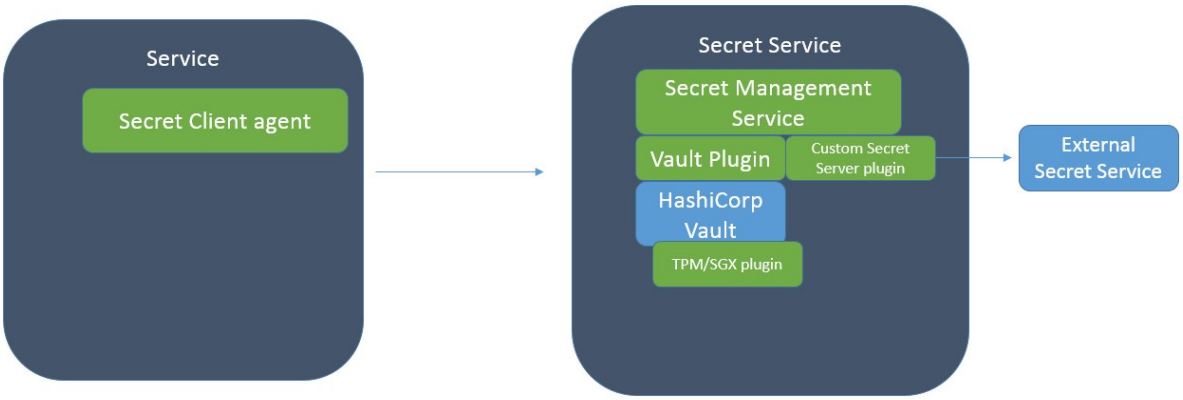
- The AAF CA server will use the TPM/SGX plugin to store private keys in hardware.
- The hardware TPM/SGX will also be used for private key related operations by AAF.



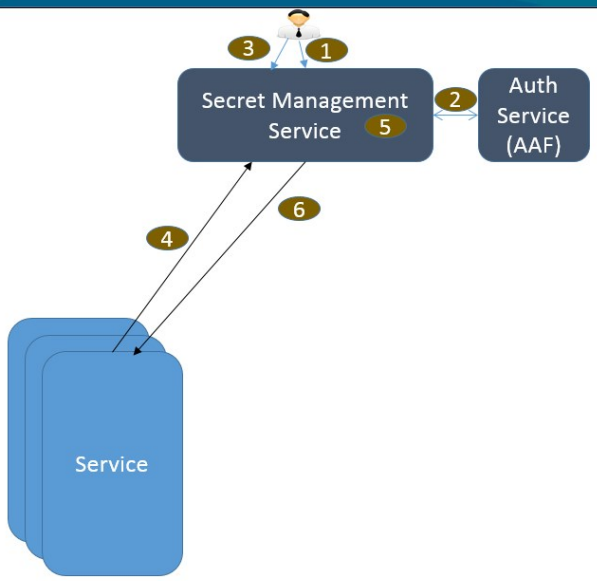
ONAP
OPEN NETWORK AUTOMATION PLATFORM

Secret Management Service

Secret Service: Architecture Blocks



Secret Server – High level flow in ONAP



It is expected that Certificate management and Auth services are brought up first.

Creating Secret Domain

1. User creates authentication session by passing username/password OR grant token given by Auth Service.
2. SMS validates the User credentials using Auth Service and if they have the right permissions.
3. User instructs SMS to create secret domain (Policies, CA-Cert, Set of Subject name prefix to allow vs permissions)

Service instance bring up (Assumes that service already got the certificate provisioned)

4. Service makes Secret service request via TLS. (Create Secret/Read Secret)
5. SMS validates the certificate credentials, if valid and if policy allows it, perform the operation. Also, create and return SMS-token for future operations.
6. SMS returns the results

Service uses secrets for service specific processing


Hardware Security in AAF.pptx