Certificate update procedure in DCAE

Note: For Frankfurt, the certificates are no longer preloaded into DCAE tls-init-container. The newer version of org.onap.dcaegen2.deployments.tls-init-container:1.2.2 (build off onap/aaf/aaf_agent:2.1.15) generates the DCAE certificate during component deployments.

For DUBLIN - DCAE service components will use common certifcates generated from AAF/test instance and made available during deployment of DCAE TLS init container.

The updated certificates should be loaded under https://git.onap.org/dcaegen2/deployments/tree/tls-init-container/tls

DCAE has generalized process of certificate distribution as documented here - https://docs.onap.org/en/latest/submodules/dcaegen2.git/docs/sections/tls_enablement.html

Setup of AAF based certificate

Note: Check validity of cert is at least 1 year from date of generation

Request Access to AAF test instance:

- Create a task ticket with components name “Multi-geo LAB” on ONAP OPENLABS JIRA requesting access to POD-ONAP-01 and OpenVPN credentials.
- Assign the ticket to Stephen Gooch (stephen.gooch@windriver.com)

Add "10.12.5.145 aaf-onap-test.osaaf.org" to your hosts file

Once the VPN is set up, you can access the AAF gui at https://aaf-onap-test.osaaf.org:8200/gui/home, use the following credentials to login:

- username: ~~aaf_admin~~ mmanager
- password: demo123456!

Once there, click My Namespaces > org.onap.dcae > Cred Details > Expand > View All > Details:

- Add your components SAN into the SANs filed (**Do not remove any existing values**)
- Set "Renewal Days before Expiration" to 364
- Click **Update**

Now you are finished with the AAF gui.

For the Frankfurt release, this is all that needs to be done. The manual steps described below have been replaced by automatic actions performed at the time a component is deployed.

Translation of the generated certificate into TLS container artifacts

Once you have updated the certificate on the AAF gui, you can create the required artifacts.

Create a folder on your local machine called AAF and enter this folder.

Pull the AAF Agent image:

```
docker pull nexus3.onap.org:10001/onap/aaf/aaf_agent:2.1.11-SNAPSHOT
```

Download the agent script

Rename it to agent.sh

Run the script and enter the following values as requested:

```
bash agent.sh bash

CADI Version (2.1.11-SNAPSHOT): <version of local AAF image pulled from docker>
Docker Repo (nexus3.onap.org:10003): nexus3.onap.org:10001
HOSTNAME (blank for Default): <default>
CONTAINER_NS (onap): <default>
AAF's FQDN: aaf-onap-test.osaaf.org <URL of AAF mentioned above>
AAF FQDN IP (10.12.6.214): <default>
Deployer's FQI: deployer@people.osaaf.org
App's Root FQDN: dcae
App's FQI (dcae@dcae.onap.org): <default>
APP's AAF Configuration Volume (dcae_config): <default>
DRIVER (local): <default>
LATITUDE of Node: 53.4239 <any acceptable lattitude value>
LONGITUDE of Node: 7.9407 <any acceptable longitude value>
```

Run the following command to view the certificate passwords:

```
agent showpass


Password: demo123456!
cadi_truststore_password=
cadi_keystore_password_jks=
cadi_key_password=
cadi_keystore_password=
cadi_keystore_password_p12=
Challenge=
```

Save these passwords, we will need them later.

Next you will need to fetch the generated artifacts from the aaf_agent container, you can do this with the following command (Run "docker ps" to get the container id)

```
docker cp <container-id>:/opt/app/osaaf/local/ ~/AAF/certs/
```

You should now have the following files on your local machine (Note: There will be some other files but we do not need them):

- org.onap.dcae.jks
- org.onap.dcae.key
- org.onap.dcae.p12
- org.onap.dcae.trust.jks

Following steps are specific to DCAE to load the generated certificate into org.onap.dcaegen2.deployments.tls-init-container

Rename these files as follows:

- org.onap.dcae.jks > cert.jks
- org.onap.dcae.key > key.pem
- org.onap.dcae.p12 > cert.p12
- org.onap.dcae.trust.jks > trust.jks

Now we need to create the password files using the passwords we seen earlier, create the following files:

- jks.pass (put the password labelled "cadi_keystore_password_jks" in here)
- p12.pass (put the password labelled "cadi_keystore_password_p12" in here)
- trust.pass (put the password labelled "cadi_truststore_password" in here)

Next, we need to create the .pem files, we can do this with the followings commands:

```
keytool -exportcert -rfc -file cacert.pem -keystore trust.jks -alias ca_local_0
Enter keystore password: <enter cadi_truststore_password here>


openssl pkcs12 -in cert.p12 -out cert.pem
Enter import password: <enter cadi_keystore_password_p12 here>
Enter PEM pass phrase: <enter cadi_keystore_password_p12 here>
Verifying - Enter PEM pass phrase: <enter cadi_keystore_password_p12 here>
```

Edit cert.pem and remove the text (bag attributes) in between the certs, so you are just left with two certs.

Finally, we will encode cert.jks, cert.p12 and trust.jks in base64:

```
base64 cert.jks > cert.jks.b64
base64 cert.p12 > cert.p12.b64
base64 trust.jks > trust.jks.b64
```

You should now have all of the required artifacts:

- cert.jks.b64
- cert.p12.b64
- trust.jks.b64
- cert.pem
- cacert.pem
- key.pem
- jks.pass
- p12.pass
- trust.pass

These artifacts must be uploaded to the TLS Container repo https://git.onap.org/dcaegen2/deployments/tree/tls-init-container/tls

Blueprint updates

Once the updated artifacts have been placed in the TLS Container repo, you will need to update your components blueprint by adding a new node_template, the cert_directory parameter is the location on your container in which you expect to find the certificates

```
tls_info:
  cert_directory: '/opt/app/component-name/etc/cert'
  use_tls: true
```

(Note that the cert_directory entry does not have a trailing /.)

Current SAN Listing

```
config-binding-service, config-binding-service.onap, config-binding-service.onap.svc.cluster.local, dcae-
cloudify-manager, dcae-cloudify-manager.onap, dcae-cloudify-manager.onap.svc.cluster.local, dcae-tca-analytics,
dcae-tca-analytics.onap, dcae-tca-analytics.onap.svc.cluster.local, dcae-ves-collector, dcae-ves-collector.
onap, dcae-ves-collector.onap.svc.cluster.local, deployment-handler, deployment-handler.onap, deployment-
handler.onap.svc.cluster.local, holmes-engine-mgmt, holmes-engine-mgmt.onap, holmes-engine-mgmt.onap.svc.
cluster.local, holmes-rule-mgmt, holmes-rules-mgmt.onap, holmes-rules-mgmt.onap.svc.cluster.local, inventory,
inventory.onap, inventory.onap.svc.cluster.local, policy-handler, policy-handler.onap, policy-handler.onap.svc.
cluster.local,dcae-hv-ves-collector, dcae-hv-ves-collector.onap, dcae-hv-ves-collector.onap.svc.cluster.local,
dcae-prh, dcae-prh.onap, dcae-prh.onap.svc.cluster.local, dcae-datafile-collector, dcae-datafile-collector.
onap, dcae-datafile-collector.onap.svc.cluster.local, dcae-pm-mapper, dcae-pm-mapper.onap, dcae-pm-mapper.onap.
svc.cluster.local, bbs-event-processor, bbs-event-processor.onap, bbs-event-processor.onap.svc.cluster.local
```