We provide examples of commands we ran as part of setting up the collective.

The examples we provide are merely illustrations of how we did it; they are not to be taken as how you must do it. You may modify commands to match your environment. All examples are provided "as is," without warranty of any kind.

Always review security commands with your security administrator.
What is a "Collective?"

A collective is ...

- A collection of Liberty servers arranged into a logical grouping for purposes of administration.
- A "controller" serves as the interface point; "members" are administered across a TCP/IP network.
- From the controller you can: see the state of all the servers; start and stop servers; and transfer files from and to members.
- Is not exclusive to z/OS; but collectives on z/OS can do unique things: start servers as STCs, use SAF for SSL and SSH.

* The controller can be arranged in a highly-available manner by grouping several servers configured as controllers into a "replica set"
You start by creating the Liberty servers you intend to include in the collective. This is done in the usual manner – "server create <server name>". It helps to have a plan for the collective mapped out ahead of time. We'll show you the plan we had for this document's collective.

The controller is created first. There is a shell script for this purpose ("collective create"). It creates some directories and files, as well as XML you add to the server.xml for the server that will be the controller.

The server that will act as the controller is then started. The controller must be active when a "join" of a member server is attempted.

You join a member to a controller's collective with the "collective join" command. That does several things: (1) communicates with the controller, (2) creates directories under the member, and (3) produces XML you add to the server.xml for the server being joined.

The member server is then started. It communicates with the controller at startup. To validate the member is properly joined, there is a message in the log output for the member.

After the collective is built, you can cycle back and push the SSL certificates into SAF keyrings, as well as move the collective-wide SSH keys into SAF keyrings.
Creating the servers ...

Logical representation

File System layout

See speaker notes for details behind each numbered circle on chart
Creating the Initial Servers
Create Mount Directory and File System

1. `mkdir /shared/lzcollect`

This was the location we planned for the root of our collective.

2. `DEFINE CLUSTER (NAME(OMVS.LZCOLL.HOME.ZFS) - LINEAR CYL(50 10) SHAREOPTIONS(3))`

This is the JCL we used to allocate the ZFS we mounted at /shared/lzcollect. The size of this ZFS was arbitrarily set at 50 cylinders, which was far more than is needed to hold the initial collective files.

3. `MOUNT FILESYSTEM('OMVS.LZCOLL.HOME.ZFS') MOUNTPOINT('/shared/lzcollect') TYPE(ZFS) MODE(RDWR)`

The MOUNT command mounted the ZFS in R/W mode at the directory location. The ownership and file permissions we set later. You should consider updating BPXPARMS to mount this at time of IPL.

See speaker notes for details behind each numbered circle on chart
Create WLP_USER_DIR Locations and OMVS Home Directory

Create the servers...

Controller's WLP_USER_DIR

```
#!/bin/sh
mkdir /shared/lzcollect/controller
mkdir /shared/lzcollect/home
mkdir /shared/lzcollect/home/LZANGL
mkdir /shared/lzcollect/home/LZCOLLCT
mkdir /shared/lzcollect/home/LZCONTRL
mkdir /shared/lzcollect/home/LZMEMBER
 mkdir /shared/lzcollect/members
```

OMVS ID home dirs

```
mkdir /shared/lzcollect/home/LZANGL
mkdir /shared/lzcollect/home/LZCOLLCT
mkdir /shared/lzcollect/home/LZCONTRL
mkdir /shared/lzcollect/home/LZMEMBER
```

Members' WLP_USER_DIR

```
mkdir /shared/lzcollect/members
```

Note: ownership and permissions not important at this point. We intend to 'chown' and 'chmod' everything to its proper settings after we have everything built.

An example of a shell script to automate this process

See speaker notes for details behind each numbered circle on chart
An example of a shell script to automate this process

```bash
#!/bin/sh

cd /usr/lpp/zWebSphere/18002/bin
export JAVA_HOME=/shared/java/J8.0_64
export WLP_USER_DIR=/shared/lzcollect/controller
./server create LZCONT
export WLP_USER_DIR=/shared/lzcollect/members
./server create LZMEMC
./server create LZMEMD

echo 'All Done!'
```

See speaker notes for details behind each numbered circle on chart
Update server.xml and server.env Files

Controller LZCONT

<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="9080" httpsPort="9443" />

Controller LZMEMC

<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="10080" httpsPort="10443" />

Controller LZMEMD

<httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="10080" httpsPort="10443" />

Controller LZCONT

WLP_SKIP_MAXPERMSIZE=true
WLP_SKIP_MAXPERMSIZE=true
JAVA_HOME=/shared/java/J8.0_64

Controller LZMEMC

WLP_SKIP_MAXPERMSIZE=true
WLP_SKIP_MAXPERMSIZE=true
JAVA_HOME=/shared/java/J8.0_64

Controller LZMEMD

WLP_SKIP_MAXPERMSIZE=true
WLP_SKIP_MAXPERMSIZE=true
JAVA_HOME=/shared/java/J8.0_64

See speaker notes for details behind each numbered circle on chart
Create the User IDs and Groups

1. **LZANGLG**
   - Liberty Angel Group ID

2. **LZANGL**
   - Liberty Angel Task ID

3. **LZCOLLCT**
   - Liberty Server File Owning ID

4. **LZCONTRL**
   - Liberty Server Controller ID

5. **LZMEMBER**
   - Liberty Server Member ID

See speaker notes for details behind each numbered circle on chart

```bash
//LZIDS JOB ...
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
ADDGROUP LZANGLG OMVS(AUTOgid) OWNER(SYS1)
ADDUSER LzAnGL DFLTGRP(LZANGLG) OMVS(AUTOUID - 
   HOME(/shared/lzcollect/home/LZANGL) - 
   PROGRAM(/bin/sh)) NAME('Liberty Angel') - 
   NOPASSWORD NOOIDCARD

ADDGROUP LZGRP OMVS(AUTOgid) OWNER(SYS1)
ADDUSER LZCONTRL DFLTGRP(LZGRP) OMVS(AUTOUID - 
   HOME(/shared/lzcollect/home/LZCONTRL) - 
   PROGRAM(/bin/sh)) NAME('Controller ID')
ADDUSER LZMEMBER DFLTGRP(LZGRP) OMVS(AUTOUID - 
   HOME(/shared/lzcollect/home/LZMEMBER) - 
   PROGRAM(/bin/sh)) NAME('Member ID')
ADDUSER LZCOLLCT DFLTGRP(LZGRP) OMVS(AUTOUID - 
   HOME(/shared/lzcollect/home/LZCOLLCT) - 
   PROGRAM(/bin/sh)) NAME('File ID')
/*
//*/
```
CHOWN and CHMOD the Directories and Files

An example of a shell script to automate this process

```bash
#!/bin/sh
chown LZCOLLCT:LZGRP /shared/lzcollect
chmod 755 /shared/lzcollect
chown -R LZCONTRL:LZGRP /shared/lzcollect/controller
chmod 700 /shared/lzcollect/home/LZANGL
chown LZANGL:LZANGLG /shared/lzcollect/home/LZANGL
chmod 700 /shared/lzcollect/home/LZANGL
chown LZCOLLCT:LZGRP /shared/lzcollect/home/LZCOLLCT
chmod 700 /shared/lzcollect/home/LZCOLLCT
chown LZCONTRL:LZGRP /shared/lzcollect/home/LZCONTRL
chmod 700 /shared/lzcollect/home/LZCONTRL
chown LZMEMBER:LZGRP /shared/lzcollect/home/LZMEMBER
chmod 700 /shared/lzcollect/home/LZMEMBER
chown -R LZMEMBER:LZGRP /shared/lzcollect/members
chmod 750 /shared/lzcollect/members
echo 'All Done!'
```

The ID that performs these commands must have the authority to chown and chmod. See your security administrator if your ID lacks that.
Creating the JCL Start Procedures in PROCLIB

/*-------------------------------------------------------
//  SET ROOT='/u/MSTONE1/wlp'
//*-------------------------------------------------------
//* Start the Liberty angel process
//*-------------------------------------------------------
//* This proc may be overwritten by fixpacks or iFixes.
//* You must copy to another location before customizing.
//*-------------------------------------------------------
//STEP1 EXEC PGM=BPXBATA2,REGION=0M,TIME=NOLIMIT,
:

/*-------------------------------------------------------
//  SET INSTDIR='/u/MSTONE1/wlp'
//*-------------------------------------------------------
//* Start the Liberty server

See speaker notes for details behind each numbered circle on chart
STARTED Profiles to Assign Task ID, and Starting Servers as z/OS STCs

//LZSTART JOB ...
//**----------------------------------------
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=* //SYSTSIN DD *
RDEFINE STARTED LZANGL.* UACC(NONE) -
   STDATA(USER(LZANGL) GROUP(LZANGLG) -
   PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

RDEFINE STARTED LZCONT.* UACC(NONE) -
   STDATA(USER(LZCONTRL) GROUP(LZANGLG) -
   PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

RDEFINE STARTED LZMEMC.* UACC(NONE) -
   STDATA(USER(LZMEMBER) GROUP(LZGRP) -
   PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

RDEFINE STARTED LZMEMD.* UACC(NONE) -
   STDATA(USER(LZMEMBER) GROUP(LZGRP) -
   PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
/**

START command    ID Assigned
Angel Task        /S LZANGL    LZANGL
Controller Task   /S LZCONT    LZCONTRL
SYSC Member Task  /S LZMEMC    LZMEMBER
SYSD Member Task  /S LZMEMD    LZMEMBER

=SDSF.DA with prefix of LZ*

See speaker notes for details behind each numbered circle on chart

Overall summary...
Summary: The Server Layout and Directory Structure

```
[Diagram showing directory structure]
```

- **/shared/lzcollect**
  - **/controller**
    - **/servers**
      - **/LZCONT**
        - Configuration files and other directories located under the LZCONT directory
  - **/home**
    - **/LZANGL**
    - **/LZCOLLCT**
    - **/LZCONTRL**
    - **/LZMEMBER**
    - **/members**
      - **/servers**
        - **/LZMEMC**
        - **/LZMEMD**
        - Configuration files and other directories located under the LZMEMC and LZMEMD directories
Summary: The server.xml and server.env Files for LZCONT

Location: /shared/lzcollect/controller/servers/LZCONT

server.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

<!-- Enable features -->
<featureManager>
  <feature>jsp-2.3</feature>
</featureManager>

<!-- To access this server from a remote client add a host attribute to the following element, e.g. host="*" -->
<httpEndpoint id="defaultHttpEndpoint"

host="*"
httpPort="9080"
httpsPort="9443" />

<!-- Automatically expand WAR files and EAR files -->
<applicationManager autoExpand="true"/>
</server>
```

We added the `host="*"` so a browser using a host name other than "localhost" could access this server. The port values were the default.

server.env

```
WLP_SKIP_MAXPERMSIZE=true
JAVA_HOME=/shared/java/J8.0_64
```

The file and the first two properties were created when the server was created. We added the JAVA_HOME value.
Summary: The server.xml and server.env Files for LZMEMC and LZMEMD

**Location:** /shared/lzcollect/members/servers/LZMEMC
/shared/lzcollect/members/servers/LZMEMD

**server.xml**

```xml
<?xml version="1.0" encoding="UTF-8"?><server description="new server">
    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.3</feature>
    </featureManager>
    <!-- To access this server from a remote client add a host attribute to the following element, e.g. host="*" -->
    <httpEndpoint id="defaultHttpEndpoint" host="*" httpPort="10080" httpsPort="10443"/>
    <!-- Automatically expand WAR files and EAR files -->
    <applicationManager autoExpand="true"/>
</server>
```

We added the **host="*"** so a browser using a host name other than "localhost" could access this server. We updated the port values to 10080 and 10443 as shown.

**server.env**

- `WLP_SKIP_MAXPERMSIZE=true`
- `JAVA_HOME=/shared/java/J8.0_64`
- `WLP_ZOS_PROCEDURE=LZMEMx`

The file and the first two properties were created when the server was created. We added the **JAVA_HOME** and **WLP_ZOS_PROCEDURE** values.
Creating the Starting the Controller Server
Two Approaches to the ID used for 'create controller'

1 - Give the Controller ID a password and log into Telnet shell environment with that ID

ALTUSER LZCONTRL PASSWORD(LZCONTRL) NOEXPIRED

ALTUSER LZCONTRL NOPASSWORD

2 - Use SAF SURROGAT and 'su' to the ID from another ID

RDEFINE SURROGAT BPX.SRV.LZCONTRL
RDEFINE SURROGAT LZCONTRL.SUBMIT
PERMIT BPX.SRV.LZCONTRL CLASS(SURROGAT) ID(BAGWELL) ACC(READ)
PERMIT LZCONTRL.SUBMIT CLASS(SURROGAT) ID(BAGWELL) ACC(READ)
SETROPTS RACLIST(SURROGAT) REFRESH

> su - s LZCONTRL
> whoami
> LZCONTRL
> exit
> whoami
> BAGWELL
The 'collective create' Command, Artifacts Created, and the ID Used to Execute Command

The server named on the 'collective create' command. This turns the server into a controller.

XML is also produced by the command, which you use to update the server.xml.

The 'collective create' command will create these directories and files under the server named on the command.

Sequence of activities ...

1. ./collective create LZCONT --keystorePassword=password

See speaker notes for details behind each numbered circle on chart
Sequence of Activities to turn LZCONT into a Controller

1. ALTER the LZCONTRL ID and give it a password

   ALTUSER LZCONTRL PASSWORD(LZCONTRL) NOEXPIRED
   This is only temporary. The controller's STC ID does not need a password for the server to operate as a controller.

2. Open a Telnet session and log in as ID

   To make things as simple as possible, session should be opened on the system where the controller will operate. An OMVS window may be more challenging. You won't be able to log into LZCONTRL in TSO, so you won't be able to get to OMVS as LZCONTRL. You can 'su' to LZCONTRL in OMVS if you have SAF SURROGAT established.

   export JAVA_HOME=/shared/java/J8.0_64
   export WLP_USER_DIR=/shared/lzcollect/controller
   cd /usr/lpp/zWebSphere/18003/bin

   ./.collective create LZCONT --keystorePassword=password
   The command will create a set of XML that is piped to the Telnet session. You can pipe it to a file using the --createConfigFile=file operand. If to screen, then copy and paste for updating the server's server.xml file.

3. Set environment variables, then change to the /bin directory

4. Issue the 'create' command and capture screen output

5. ALTER the ID back to NOPASSWORD

   ALTUSER LZCONTRL NOPASSWORD
   It is a good practice to remove passwords from IDs that do not need to have them.
The Copy-and-Paste XML Produced, Part 1 of 2

The output for a controller was more than could be fit on one chart and still be readable...

In addition to creating a set of directories and files under the server to act as the controller, the 'collective create' command produces a set of XML that you use to update the server.xml file.

```xml
<featureManager>
  <feature>collectiveController-1.0</feature>
</featureManager>

<variable name="defaultHostName" value="wsc3" />

<!-- Define the host name for use by the collective. If the host name needs to be changed, the server should be removed from the collective and re-joined or re-replicated. -->

<quickStartSecurity userName="" userPassword="" />

<!-- clientAuthenticationSupported set to enable bidirectional trust -->
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore" trustStoreRef="defaultTrustStore" clientAuthenticationSupported="true" />

(see next chart for the rest of the XML that was produced)

It's a good idea to copy/paste in the comments as well so anyone inspecting the XML later has an idea what these elements are for.

Important! This requires you to update the userName= and userPassword= values. See speaker notes for details.

See speaker notes for details behind each numbered circle on chart.
The Copy-and-Paste XML Produced, Part 2 of 2

The output for a controller was more than could be fit on one chart and still be readable ...

(see previous chart for the first part of the XML that was produced)

<!-- inbound (HTTPS) keystore -->
<keyStore id="defaultKeyStore" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/security/key.jks" />

<!-- inbound (HTTPS) truststore -->
<keyStore id="defaultTrustStore" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/security/trust.jks" />

<!-- server identity keystore -->
<keyStore id="serverIdentity" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/collective/serverIdentity.jks" />

<!-- collective trust keystore -->
<keyStore id="collectiveTrust" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/collective/collectiveTrust.jks" />

<!-- collective root signers keystore -->
<keyStore id="collectiveRootKeys" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/collective/rootKeys.jks" />

It's a good idea to copy/paste in the comments as well so anyone inspecting the XML later has an idea what these elements are for.
Start the Server and Check messages.log for Success Messages

/S LZCONT  Command to start the controller server

CWWKE0001I: The server LZCONT has been launched.
  (messages removed from log file to save space)
  :
  CWWKX1000I: The SingletonMessenger MBean is available.
  CWWKX9079I: The CollectiveElectorFactory is available.
  CWWKX9000I: The CollectiveRepository MBean is available.
  CWWKX8116I: The server STARTED state was successfully published to the collective repository.
  CWWKX8114I: The server's paths were successfully published to the collective repository.
  CWWKX9065I: The ControllerConfig MBean is available.
  CWWKX9069I: The AdminMetadataManager MBean is available.
  CWWKX1002I: Singleton service FailoverSingletonService for scope collective is created.
  CWWKX7912I: The FileServiceMXBean attribute ReadList was successfully updated.
  CWWKX7912I: The FileServiceMXBean attribute WriteList was successfully updated.
  CWWKX9030I: The ClusterManager MBean is available.
  CWWKX7200I: The ServerCommands MBean is available.
  CWWKX9049I: The RepositoryConfiguration MBean is available.
  CWWKX9003I: CollectiveRegistration MBean is available.
  CWWKX8122I: The default SSH key-based configuration is being used for the host authentication configuration.
  CWWKX8154I: The remote host authentication for this server has been configured with the collective-wide SSH key.
  CWWKX7240I: The DeployService is available.
  CWWKX8112I: The server's host information was successfully published to the collective repository.
Joining and Starting the Member Servers
Sequence of Activities to join LZMEMC as a Member

**ALTER the LZMEMBER ID and give it a password**

```
ALTUSER LZMEMBER PASSWORD(LZMEMBER) NOEXPIRED
This is only temporary. The member's STC ID does not need a password for the server to operate as a member.
```

To make things as simple as possible, session should be opened on the system where the member will operate.

```
An OMVS window may be more challenging. You won't be able to log into LZCONTRL in TSO, so you won't be able to get to OMVS as LZCONTRL. You can 'su' to LZCONTRL in OMVS if you have SAF SURROGAT established.
```

**Open a Telnet session and log in as ID**

```
export JAVA_HOME=/shared/java/J8.0_64
export WLP_USER_DIR=/shared/lzcollect/members
cd /usr/lpp/zWebSphere/18002/bin
./collective join LZMEMC --host=wsc3 --port=9443
   --user=admin --password=adminpw --keystorePassword=password
```

```
1
```

The command will create a set of XML that is piped to the Telnet session. You can pipe it to a file using the
```
--createConfigFile=file operand. If to screen, then copy and paste for updating the server's server.xml file.
```

```
2
```

```
3
```

Not yet ... we still had to join the LZMEMD member, and that operated with the same STC ID of 'LZMEMBER.' So we left the ID with a password until LZMEMD was joined.

**Set environment variables, then change to the /bin directory**

**Issue the 'create' command and capture screen output**

**ALTER the ID back to NOPASSWORD?**

See speaker notes for details behind each numbered circle on chart
The Copy-and-Paste XML Produced and Used to Update server.xml

```xml
<featureManager>
  <feature>collectiveMember-1.0</feature>
</featureManager>

<!-- Define the host name for use by the collective. If the host name needs to be changed, the server should be removed from the collective and re-joined or re-replicated. -->
<variable name="defaultHostName" value="wsc3" />

<!-- Connection to the collective controller -->
<collectiveMember controllerHost="wsc3"
  controllerPort="9443" />

<!-- clientAuthenticationSupported set to enable bidirectional trust -->
<ssl id="defaultSSLConfig"
  keyStoreRef="defaultKeyStore"
  trustStoreRef="defaultTrustStore"
  clientAuthenticationSupported="true" />

<!-- inbound (HTTPS) keystore -->
<keyStore id="defaultKeyStore" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/security/key.jks" />

<!-- inbound (HTTPS) truststore -->
<keyStore id="defaultTrustStore" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/security/trust.jks" />

<!-- server identity keystore -->
<keyStore id="serverIdentity" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/collective/serverIdentity.jks" />

<!-- collective truststore -->
<keyStore id="collectiveTrust" password="{xor}Lz4sLCgwLTs="
  location="${server.config.dir}/resources/collective/collectiveTrust.jks" />
```

The collectiveMember-1.0 feature was added to the <featureManager> list in the server.xml file.

The rest was pasted into the server.xml.
Start the Server and Check messages.log for Success Messages

/S LZMEMC ← Command to start the member server

In the LZMEMC member server messages.log:

```
CWWKE0001I: The server LZMEMC has been launched.

(messages removed from log file to save space)

CWWKX0230I: The collective member opened JMX client to the collective controller: wsc3:9443
CWWKX8055I: The collective member has established a connection to the collective controller.
CWWKX8122I: The default SSH key-based configuration is being used for the host authentication configuration.
CWWKX1000I: The SingletonMessenger MBean is available.
CWWKX8116I: The server STARTED state was successfully published to the collective repository.
CWWKX8154I: The remote host authentication for this server has been configured with the collective-wide SSH key.
CWWKX8114I: The server's paths were successfully published to the collective repository.
CWWKX7912I: The FileServiceMXBean attribute ReadList was successfully updated.
CWWKX7912I: The FileServiceMXBean attribute WriteList was successfully updated.
CWWKX8112I: The server's host information was successfully published to the collective repository.
```

In the LZCONT controller server messages.log:

```
CWWKX9076I: The collective member LZMEMC on host wsc3 with the user directory %2Fshared%2Flzcollect%2Fmembers connected to the collective controller.
```
Sequence of Activities to join LZMEMD as a Member

- ALTER the LZMEMBER ID and give it a password?
- Open a Telnet session and log in as ID
- Set environment variables, then change to the /bin directory
- Issue the 'create' command and capture screen output
- ALTER the ID back to NOPASSWORD

We already did this for joining LZMEMC. Since LZMEMC and LZMEMD share the same ID, we can use that ID again when joining LZMEMD to the collective.

To make things as simple as possible, we opened the Telnet session to the SYSD LPAR, since that's where the LZMEMD member would run. That allowed the 'collective join' command to pick up the proper host name ('wsc4').

```
export JAVA_HOME=/shared/java/J8.0_64
export WLP_USER_DIR=/shared/lzcollect/members
cd /usr/lpp/zWebSphere/18002/bin
./collective join LZMEMD --host=wsc3 --port=9443
   --user=admin --password=adminpw --keystorePassword=password
ALTUSER LZMEMBER NOPASSWORD
```

This process was essentially identical to joining LZMEMC, except we executed the join command from the SYSD LPAR.

The command will create a set of XML that is piped to the Telnet session. You can pipe it to a file using the `--createConfigFile=file` operand. If to screen, then copy and paste for updating the server's server.xml file.

ALTUSER LZMEMBER NOPASSWORD
It is a good practice to remove passwords from IDs that do not need to have them.
Check for 'StrictModes' Compliance

"StrictModes" is a function of OpenSSH that enforces strict permission bits on certain directories and files. SSH won't work properly if StrictModes is in effect but the directories and files modes are too permissive.


```
/shared/lzcollect
  /controller
  /servers
    /LZCONT
    /resources
      /security
        /ssh
          id_rsa
          id_rsa.pub
        Need: 700, Was: 600
        Need: 600, Was: 600
  /home
    /LZCONTRL
    /.ssh
      Need: 700, Was: 700
      Need: 700, Was: 755
      Corrective Action: chmod 700 <dir>
      Need: 600, Was: 600
      Need: 600, Was: 600
```

See speaker notes for details behind each numbered circle on chart
Enabling the Admin Center and Using the "Quick Start Security" Credentials

The Admin Center provides an additional level of validation of collective operations. With the "quick start security" in place from the 'collection create' process, enabling the Admin Center was relatively simple:

The controller's server.xml file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
  <!-- Enable features -->
  <featureManager>
    <feature>jsp-2.3</feature>
    <feature>collectiveController-1.0</feature>
    <feature>adminCenter-1.0</feature>
  </featureManager>

  <!-- Define the host name for use by the collective. If the host name needs to be changed, the server should be removed from the collective and re-joined or re-replicated. -->
  <variable name="defaultHostName" value="wsc3" />

  <!-- TODO: Set the security configuration for Administrative access -->
  <quickStartSecurity userName="admin" userPassword="adminpw" />

  (remainder of server.xml not shown)
```

This one feature is all that's needed to enable the Admin Center.

The `<quickStartSecurity>` element was part of the 'collective create' process. This provided us the ID and password we used to log onto the Admin Center.
Accessing the Admin Center

https://wsc3.washington.ibm.com:9443/adminCenter/

1. <quickStartSecurity userName="admin" userPassword="adminpw" />

2. User Name = admin
   Password = adminpw

3. See speaker notes for details behind each numbered circle on chart

Viewing servers and status ...
Viewing the Servers and Server Status

Start LZMEMC from AdminCenter ...

See speaker notes for details behind each numbered circle on chart
Starting LZMEMC from the AdminCenter

**Note:** we stopped the servers manually before attempting an AdminCenter start operation. See speaker notes.

```
START LZMEMC,PARMS='LZMEMC -scriptLaunch'
IRR812I PROFILE LZMEMC.* (G) IN THE STARTED CLASS WAS USED 764
TO START LZMEMC WITH JOBNAME LZMEMC.
IEF695I START LZMEMC WITH JOBNAME LZMEMC IS ASSIGNED TO USER LZMEMBER, GROUP LZGRP
+CWWKE0001I: The server LZMEMC has been launched.
+CWWKF0011I: The server LZMEMC is ready to run a smarter planet.
```

**System Log**

```
START LZMEMC,PARMS='LZMEMC -scriptLaunch'
IRR812I PROFILE LZMEMC.* (G) IN THE STARTED CLASS WAS USED 764
TO START LZMEMC WITH JOBNAME LZMEMC.
IEF695I START LZMEMC WITH JOBNAME LZMEMC IS ASSIGNED TO USER LZMEMBER, GROUP LZGRP
+CWWKE0001I: The server LZMEMC has been launched.
+CWWKF0011I: The server LZMEMC is ready to run a smarter planet.
```

**Display Active**

```
  JOBNAME  StepName  SysName  Owner
  LZCONT   LZCONT    SYSC     LZCONTRL
  LZMEMC   LZMEMC    SYSC     LZMEMBER
```

See speaker notes for details behind each numbered circle on chart
Stopping LZMEMC and LZMEMD from the AdminCenter

Display Active

<table>
<thead>
<tr>
<th>JOBNAME</th>
<th>StepName</th>
<th>SysName</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>LZCONT</td>
<td>LZCONT</td>
<td>SYSC</td>
<td>LZCONTRL</td>
</tr>
</tbody>
</table>

SAF for SSL and registry/authorization ...
Using SAF Keyrings for SSL Certificates

As well as SAF for user registry and role authorization for Admin Center access
The Plan for SSL Certificates


Reminder: review all security commands with your security administrator

Connect 7

Connect 8

See speaker notes for details behind each numbered circle on chart
RACF Commands to Generate CA and Server Certificates (Part 1)

//LZCERTS JOB ...
//STEP1 EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=2M
//SYSTSPRT DD SYSPUT=*  
//SYSTSIN DD *

RACDCERT CERTAUTH GENCERT SUBJCTSDN(CN('CONTROLLER ROOT') -
   OU('LZCollective')) SIZE(2048) WITHLABEL('LZCollect.CA') -
   TRUST NOTAFTER(DATE(2031/12/31))

RACDCERT ID(LZCONTRL) GENCERT SUBJCTSDN(CN('wsc*') -
   O('IBM') OU('LZCollective')) WITHLABEL('LZControl.CERT') -
   SIGNWITH(CERTAUTH LABEL('LZCollect.CA')) SIZE(2048) -
   NOTAFTER(DATE(2031/12/30))

RACDCERT ID(LZMEMBER) GENCERT SUBJCTSDN(CN('wsc*') -
   O('IBM') OU('LZCollective')) WITHLABEL('LZMember.CERT') -
   SIGNWITH(CERTAUTH LABEL('LZCollect.CA')) SIZE(2048) -
   NOTAFTER(DATE(2031/12/30))

RACDCERT ADDRING(LZCONTRL.Ring) ID(LZCONTRL)
RACDCERT ADDRING(LZMEMBER.Ring) ID(LZMEMBER)

RACDCERT CONNECT(CERTAUTH LABEL('LZCollect.CA') -
   RING(LZCONTRL.Ring)) ID(LZCONTRL)
RACDCERT CONNECT(ID(LZCONTRL) LABEL('LZControl.CERT') -
   RING(LZCONTRL.Ring) DEFAULT) ID(LZCONTRL)

RACDCERT CONNECT(CERTAUTH LABEL('LZCollect.CA') -
   RING(LZMEMBER.Ring)) ID(LZMEMBER)
RACDCERT CONNECT(ID(LZMEMBER) LABEL('LZMember.CERT') -
   RING(LZMEMBER.Ring) DEFAULT) ID(LZMEMBER)

SETR RACLIST(DIGTCERT) REFRESH

The CA certificate used to sign the server certificates.

The controller server certificate, with OU=LZCollective.

The member server certificate, with OU=LZCollective.

The two keyrings; one for controller, one for member.

Connected the CA and server certificate to the controller ring.

Connected the CA and server certificate to the member ring.

Performed a refresh on DIGTCERT.

Reminder: review all security commands with your security administrator

More of this job on the next chart ...
RACF Commands to Generate CA and Server Certificates (Part 2)

More of this job on the previous chart ...

 Grant the controller ID the ability to list certificates and rings.

 Grant the member ID the ability to list certificates and rings.

List each keyring to validate the proper certificates are attached to each ring.

Reminder: review all security commands with your security administrator
Update to LZCONT server.xml to support SSL certificates in SAF Keyrings

1. `<feature>transportSecurity-1.0</feature>`
   - (some XML not shown here to save space on the chart)

2. `<collectiveCertificate rdn="OU=LZCollective"></collectiveCertificate>`
   - (some XML not shown here to save space on the chart)

3. <!-- inbound (HTTPS) keystore -->
   `<keyStore id="defaultKeyStore" password="{xor}Lz4sLCgwLTs="
   location="${server.config.dir}/resources/security/key.jks" />
   location="safkeyring:///LZCONTRL.Ring" password="password"
   type="JCERACFKS" fileBased="false" readOnly="true" />

4. <!-- inbound (HTTPS) truststore -->
   `<keyStore id="defaultTrustStore" password="{xor}Lz4sLCgwLTs="
   location="${server.config.dir}/resources/security/trust.jks" />
   location="safkeyring:///LZCONTRL.Ring" password="password"
   type="JCERACFKS" fileBased="false" readOnly="true" />

5. <!-- server identity keystore -->
   `<keyStore id="serverIdentity" password="{xor}Lz4sLCgwLTs="
   location="${server.config.dir}/resources/collective/serverIdentity.jks" />
   location="safkeyring:///LZCONTRL.Ring" password="password"
   type="JCERACFKS" fileBased="false" readOnly="true" />

6. <!-- collective trust keystore -->
   `<keyStore id="collectiveTrust" password="{xor}Lz4sLCgwLTs="
   location="${server.config.dir}/resources/collective/collectiveTrust.jks" />
   location="safkeyring:///LZCONTRL.Ring" password="password"
   type="JCERACFKS" fileBased="false" readOnly="true" />

7. <!-- collective root signers keystore -->
   `<keyStore id="collectiveRootKeys" password="{xor}Lz4sLCgwLTs="
   location="${server.config.dir}/resources/collective/rootKeys.jks" />
   location="${server.config.dir}/resources/collective/rootKeys.jks" />
   (some XML not shown here to save space on the chart)"}

See speaker notes for details behind each numbered circle on chart
Update to LZMEMC and LZMEMD server.xml to support SSL certificates in SAF Keyrings

1. `<feature>transportSecurity-1.0</feature>
   : (some XML not shown here to save space on the chart)
2. `<collectiveCertificate rdn="OU=LZCollective"></collectiveCertificate>
   : (some XML not shown here to save space on the chart)
3. <!-- inbound (HTTPS) keystore -->
   <keyStore id="defaultKeyStore" password="{xor}Lz4sLCgwLTs="
   location="{${server.config.dir}/resources/security/key.jks" />
   location="safkeyring:///LZMEMBER.Ring" password="password"
   type="JCERACFKS" fileBased="false" readOnly="true" />
4. <!-- inbound (HTTPS) truststore -->
   <keyStore id="defaultTrustStore" password="{xor}Lz4sLCgwLTs="
   location="{${server.config.dir}/resources/security/trust.jks" />
   location="safkeyring:///LZMEMBER.Ring" password="password"
   type="JCERACFKS" fileBased="false" readOnly="true" />
5. <!-- server identity keystore -->
   <keyStore id="serverIdentity" password="{xor}Lz4sLCgwLTs="
   location="{${server.config.dir}/resources/collective/serverIdentity.jks" />
   location="safkeyring:///LZMEMBER.Ring" password="password"
   type="JCERACFKS" fileBased="false" readOnly="true" />
6. <!-- collective truststore -->
   <keyStore id="collectiveTrust" password="{xor}Lz4sLCgwLTs="
   location="{${server.config.dir}/resources/collective/collectiveTrust.jks" />
   location="safkeyring:///LZMEMBER.Ring" password="password"
   type="JCERACFKS" fileBased="false" readOnly="true" />

See speaker notes for details behind each numbered circle on chart
Validation of SAF-based Keyrings and Certificates

Start the Controller from z/OS and log into the Admin Center (using 'admin' and 'adminpw' quick start security credentials)

The browser still issued a security warning about the certificate authority … that was expected. Our RACF CA was technically valid, but it was not a "well known" CA to the browser.

We checked the certificate and saw that the "Issued to" showed "wsc*", which is what we set in our RACF job, and the certificate was "Issued by" a CA with CN=CONTROLLER ROOT, which matched what we set in our RACF job.

In summary: this was our SAF-based certificate and not the file-based certificate from before. We were using SAF certificates.

Start the LZMEMC and LZMEMD from the Admin Center and check messages.log

In LZCONT:
- CWWKX9076I: The collective member LZMEMC on host wsc3 … connected to the collective controller.
- CWWKX9076I: The collective member LZMEMD on host wsc4 … connected to the collective controller.

In LZMEMC:
- CWWKX8112I: The server's host information was successfully published to the collective repository.

In LZMEMD:
- CWWKX8112I: The server's host information was successfully published to the collective repository.
The SERVER profiles grant the server access to z/OS authorized services, via the Angel. SAF EJBROLE access requires authorized access.

We created an ID in SAF that served as the administrative user, rather than the "admin" ID defined to "quick start security."

We created an ID in SAF that served as the thread identity for the brief period between request arrival and authentication.

The APPL profile protects access to application resources.

The security prefix is used to qualify the role definitions. We used the default profile prefix of BBGZDFLT.

The EJBROLE profiles define what authority the authenticated ID has once permitted into the application: that is, "administrator."
Defining Admin User, Unauthenticated User, Security Prefix (Part 1 of 2)

```
//LZROLES JOB ...
//*---------------------------------------------------------*
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSIN DD SYSOUT=* 
//SYSTSIN DD *
ADDUSER FRED DFLTGRP(SYS1) OMVS(AUTOUID HOME(/u/fred) -
          PROGRAM(/bin/sh)) NAME('USER FRED')
ALTUSER FRED PASSWORD(FREDSAF) NOEXPIRED
ADDGROUP WSGUESTG OMVS(AUTOgid) OWNER(SYS1)
ADDUSER WSGUEST RESTRICTED DFLTGRP(WSGUESTG) OMVS(AUTOUID -
          HOME(/u/wsguest) PROGRAM(/bin/sh)) -
          NAME('UNAUTHENTICATED USER') NOPASSWORD NOOIDCARD
RDEFINE APPL BBGZDFLT UACC(NONE) OWNER(SYS1)
PERMIT BBGZDFLT CLASS(APPL) RESET
PERMIT BBGZDFLT CLASS(APPL) ACCESS(READ) ID(WSGUEST)
RALTER APPL BBGZDFLT UACC(READ)
SETROPTS RACLIST(APPL) REFRESH:
```

More of this job on the next chart ...

The ADDUSER command created the ID of 'FRED', which was to be the administrator ID. The ALTUSER granted the ID a password of 'FREDSAF'.

The WSGUESTG group was used as the group for the "unauthenticated user." WSGUEST was the unauthenticated user. It has very little authority. It is used for that brief period of time between request and when requester is properly authenticated.

The unauthenticated ID needs READ to the APPL profile that matches the security prefix.

Reminder: review all security commands with your security administrator
Defining Admin User, Unauthenticated User, Security Prefix (Part 2 of 2)

More of this job on the previous chart ...

```
RDEFINE SERVER BBG.SECPFX.BBGZDFLT UACC(NONE)

PERMIT BBG.SECPFX.BBGZDFLT -
    CLASS(SERVER) ACCESS(READ) ID(LZCONTRL)

SETROPTS RACLIST(SERVER) REFRESH

RDEFINE EJBROLE BBGZDFLT.allAuthenticatedUsers -
    OWNER(SYS1) UACC(NONE)

PERMIT BBGZDFLT.allAuthenticatedUsers -
    CLASS(EJBROLE) ID(FRED) ACCESS(READ)

RDEFINE EJBROLE BBGZDFLT.Administrator -
    OWNER(SYS1) UACC(NONE)

PERMIT BBGZDFLT.Administrator -
    CLASS(EJBROLE) ID(FRED) ACCESS(READ)

SETROPTS RACLIST(EJBROLE) REFRESH
/*
 /**
```

- This defines the profile prefix (BBGZDFLT) and grants the controller's ID (LZCONTRL) READ access to the SERVER profile.
- This defines an EJBROLE for all authenticated users, with FRED (the intended administrator) granted READ.
- This defines an EJBROLE for the administrator role. FRED is granted READ to this, making FRED an administrator.

Reminder: review all security commands with your security administrator

See speaker notes for details behind each numbered circle on chart
SAF SERVER Profiles for Access to Authorized Services

//LZSERVER JOB ...
//*-----------------------------------------------------------------*
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
RDEFINE SERVER BBG.ANGEL.LZANGL -
    UACC(NONE) OWNER(SYS1)
PERMIT BBG.ANGEL.LZANGL CLASS(SERVER) -
    ACCESS(READ) ID(LZCONTRL)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM -
    UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM CLASS(SERVER) -
    ACCESS(READ) ID(LZCONTRL)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.SAFCRED -
    UACC(NONE) OWNER(SYS1)
PERMIT BBG.AUTHMOD.BBGZSAFM.SAFCRED CLASS(SERVER) -
    ACCESS(READ) ID(LZCONTRL)

SETROPTS RACLIST(SERVER) REFRESH
/*
/*/ 

Reminder: review all security commands 
with your security administrator

The BBG.ANGEL.LZANGL profile is specific to the "named Angel" we created (LZANGL, with NAME='LZANGL" on the PROC statement. The controller ID (LZCONTRL) is granted READ to this, which is the basic first step in access to authorized services.

The BBG.AUTHMOD.BBGZSAFM profile is another essential requirement to gaining access to authorized services. The controller ID (LZCONTRL) is granted READ to this as well.

The BBG.AUTHMOD.BBGZSAFM.SAFCRED profile is what grants the controller access to the authorized service related to checking role authorizations.
Update to Controller's server.xml to Support SAF Authentication / Authorization

<!-- Enable features -->
<featureManager>
  <feature>jsp-2.3</feature>
  <feature>collectiveController-1.0</feature>
  <feature>adminCenter-1.0</feature>
  <feature>transportSecurity-1.0</feature>
  <feature>zosSecurity-1.0</feature>
</featureManager>

<!-- Define the host name for use by the collective. 
If the host name needs to be changed, the server should be
removed from the collective and re-joined or re-replicated. -->
<variable name="defaultHostName" value="wsc3" />

<!-- TODO: Set the security configuration for Administrative access -->
<quickStartSecurity userName="admin" userPassword="adminpw" />

<safAuthorization id="saf" racRouteLog="ASIS" />
<safCredentials profilePrefix="BBGZDFLT" unauthenticatedUser="WSGUEST" />
<safRegistry id="saf" realm="WASRealm" />
<safRoleMapper profilePattern="%profilePrefix%.%role%" />

<collectiveCertificate rdn="OU=LZCollective"/>

<!-- clientAuthenticationSupported set to enable bidirectional trust -->
: (some XML not shown here to save space on the chart)

The zosSecurity-1.0 feature was added. This enables the feature that provides the ability to access z/OS SAF for, among other things, role authorizations.

The <quickStartSecurity> line was removed.

See speaker notes for details behind this
Named Angels', the bootstrap.properties File, and the "Is Available" Messages

No name = default Angel

Angel
Name=' '  
Angel
Name='LZANGL'  

SERVER Profile Access

messages.log file
CWWKB0122I: This server is connected to the LZANGL angel process.
CWWKB0103I: Authorized service group KERNEL is available.
CWWKB0103I: Authorized service group SAFCREDC is available.
CWWKB0104I: Authorized service group LOCALCOM is not available.
CWWKB0104I: Authorized service group PRODMGR is not available.
CWWKB0104I: Authorized service group TXRRS is not available.
CWWKB0104I: Authorized service group WOLA is not available.
CWWKB0104I: Authorized service group ZOSAIO is not available.
CWWKB0104I: Authorized service group ZOSDUMP is not available.
CWWKB0104I: Authorized service group ZOSWLM is not available.
CWWKB0104I: Authorized service group CLIENT.WOLA is not available.

See speaker notes for details behind each numbered circle on chart

SERVER Profile Access

com.ibm.ws.zos.core.angelName=LZANGL
com.ibm.ws.zos.core.angelRequired=true

We created this file and added these lines

See speaker notes for details behind each numbered circle on chart
Validation of SAF for Authentication and Authorization

https://wsc3.washington.ibm.com:9443/adminCenter/

"Login error: unrecognized user name and/or password"
Using SAF Keyrings for SSH Keys
Overview of SSH Keys in SAF

1. Add Ring
2. Generate Keys
3. Connect

server.xml

```xml
<collectiveHostAuthInfo
  safKeyring="safkeyring:///LZCONTRL.SSHRing"
  safCertificateLabel="LZCollectiveSSH" />

<keyStore id="collectiveRootKeys" password="{xor}Lz4sLCgwL7s="
  location="${server.config.dir}/resources/collective/rootKeys.jks" />
```

See speaker notes for details behind each numbered circle on chart.
Generating the SSH Keys

//LZSSH JOB...
//------------------------------------------------------------*
//RACF EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=* 
//SYSTSIN DD *
RACDCERT ID(LZCONTRL) GENCERT SUBJECTSDN(CN('LZCollective') -
O('IBM') C('US')) SIZE(2048) WITHLABEL('LZCollectiveSSH')
RACDCERT ADDRING(LZCONTRL.SSHRing) ID(LZCONTRL)
RACDCERT CONNECT(ID(LZCONTRL) LABEL('LZCollectiveSSH') -
RING(LZCONTRL.SSHRing)) ID(LZCONTRL)
SETR RACLST(DIGTCERT) REFRESH
RACDCERT ID(LZCONTRL) LISTRING(LZCONTRL.SSHRing)
/*
/*

The RACDCERT command generates a 2048 bit RSA key pair. Note the label: this
is referenced in the server.xml update.

A new ring is created called LZCONTRL.SSHRing.

The SSH key certificate is connected to the new ring.

Reminder: review all security commands with your security administrator.

Update server.xml ...
Updating the Controller's server.xml

Detour: SAF SSH at time of collective create ...

<safAuthorization id="saf" racRouteLog="ASIS" />
<safCredentials profilePrefix="BBGZDFLT" unauthenticatedUser="WSGUEST" />
<safRegistry id="saf" realm="WASRealm" />
<safRoleMapper profilePattern="%profilePrefix%.%role%" />

<collectiveCertificate rdn="OU=LZCollective"></collectiveCertificate>

<collectiveHostAuthInfo
  safKeyring="safkeyring:///LZCONTRL.SSHRing"
  safCertificateLabel="LZCollectiveSSH" />

<!-- clientAuthenticationSupported set to enable bidirectional trust -->
<ssl id="defaultSSLConfig"
  keyStoreRef="defaultKeyStore"
  trustStoreRef="defaultTrustStore"
  clientAuthenticationSupported="true" />

<!-- inbound (HTTPS) keystore -->
<keyStore id="defaultKeyStore"
  location="safkeyring:///LZCONTRL.Ring" password="password"
  type="JCERACFKS" fileBased="false" readOnly="true" />

<!-- collective trust keystore -->
<keyStore id="collectiveTrust"
  location="safkeyring:///LZCONTRL.Ring" password="password"
  type="JCERACFKS" fileBased="false" readOnly="true" />

<!-- collective root signers keystore -->
<keyStore id="collectiveRootKeys" password="{xor}Lz4sLCgwLTs=
  location="${server.config.dir}/resources/collective/rootKeys.jks" />

The <collectiveHostAuthInfo> element specifies the keyring and certificate that represents the SSH keys.

The <keyStore> element generated when the controller was initially created is no longer needed. We removed it.
Detour: SSH in SAF at Time of Collective Creation

It's possible to "bring your own" SSH keys at the time you initially build the controller with the 'collective create' command:

```
./collective create LZCONT --keystorePassword=password
   --safKeyring="safkeyring://LZCONTRL.SSHRing" --safCertificateLabel="LZCollectiveSSH"
```

The `--safKeyring` parameter specifies the keyring in SAF where the SSH keys can be found.

The `--safCertificateLabel` parameter specifies the label of the certificate to use.

**Note:** this creates the `<collectiveHostAuthInfo>` element in the XML produced to update the server.xml file. The file-based SSH key files are *not* being produced. This does not generate SAF certificate, nor does it create the keyring. That is done before the 'collective create' is issued with these parameters.
Manually Stop and Restart the Collective to Propagate New Authorized Keys Information

Before attempting to start member servers through the Admin Center, manually start the member servers so the "authorized keys" (the SSH public key) can be copied out to the "home" directory of the member servers' ID:

```
/shared/lzcollect
  /controller
    /servers
      /LZCONT
        /resources
          /security
            /ssh
              id_rsa
              id_rsa.pub
        /home
          /LZCONTRL
            /.ssh
              authorized_keys
          /LZMEMBER
            /.ssh
              authorized_keys
```

See speaker notes for details behind each numbered circle on chart.

(long string of mixed numbers and letters) … Generated on Thu May 31 11:02:22 GMT 2018
(long string of mixed numbers and letters) … Extracted on Thu May 31 11:30:04 GMT 2018
Summary
Summary

• In the first section we planned and built the servers that would be used for the collective.

• In the second section we created and started the controller from one of the servers. At that point we were using file-based SSL, file-based SSH, and "quick start security."

• In the third section we joined the member servers to the collective. We enabled the Admin Center and illustrated starting a member from the Admin Center.

• In the fourth section we used SAF for the SSL certificates, and we used SAF for authentication and authorization for the Admin Center.

• In the fifth section we used SAF for the SSH keys.

• We could have used SAF for the SSH keys at the time we created the controller. For this document we illustrated a sequential process: file-based first, then SAF-based SSH.
References

Moving SSL Certificates into SAF

Generating SSH Keys

SSH StrictModes
Document Change History

June 28, 2018   Original document.