Air-gapped upgrade handbook

Air-Gapped upgrade

The air-gapped upgrade strategy allows the upgrade of SA by migrating data and servers from an old environment to a new one. It has the advantage of leaving the existing environment intact, only disrupting managed server accessibility during the server migration phase. This upgrade technique is not documented in the officially published user manual for the SA product, as it has been developed by the Technical Success team and the tools to support it.

Pros

- Installation of a new SA environment. Hardware, OS and Database can all be upgraded during the new build.
- Failures during the new environment build do not impact the existing production system.
- The new SA environment once installed, can be QA'd and validated before migration begins.
- Fresh SA installation resets 10yr certificate event negating a mesh re-certification.
- Disruption to business only during the server migration process.

Cons

- Servers and Data have to be migrated.
- Some data loss will occur. This could be considered a Pro if a reduced database and file-system usage is the desired outcome. Audit results, Snapshot results, device history and job history, this is not an exhaustive list.
- New mesh IP addresses necessitate opening network firewall rules.
- Issues in the source mesh may prevent data migration and will require resolution before export is successful.
- Patches to the CBT data export tool in the current SA environment will be required before its used
- No formal documentation for the process as it's not an officially supported upgrade path
- Need to support old and new environments until all servers are migrated.

At a high level, the workflow is easily understood

- Build a new SA environment
- Migrate data from the old mesh to the new
- Move servers from the old mesh to the new

This is a simplistic work instruction, and, of course, the devil is in the detail. This guide will explain the detail enabling SA administrators to be comfortable performing this upgrade service in their environments.

3rd Party core certification considerations

Pre SA 10.60 agents

Starting with SA 10.60 a core may use 3rd party certificates. If the core is configured in this mode, the APX will fail if moving pre 10.60 agents as old agents won't support 3rd party mode. This is not a limitation of the APX. Its a limitation of the agents being moved. See Agent Migration APX failure with core in 3rd party certificate mode

Why won't this work?

Pre-10.60 agents only allow access to a core certificate if the issuer CN is 'opsware'. In 3rd party certificate mode, this won't be true.

Making your core issue temporary self-signed certificates will allow the agent to move but once attached to the new core, the Agent will DENY all requests due to the above CN issue.

For these reasons pre-10.60 agents cannot be migrated to a core operating in 3rd Party Certificate mode.

Allow self-signed certificates

The APX that moves servers between meshes need to use the self-signed bootstrap certificate. This means if the target mesh is running in 3rd party certificate mode it must also be configured to allow the bootstrap certificate.

```
/opt/opsware/support/bin/sa_config_checkfix -k spin.agent.bootstrap_enabled -v 1 --set
```

If this is set to 1, then a core operating in 3rd party mode can issue a temporary self-signed certificate with a limited life to managed servers, while they wait for their 3rd party certificate.

The temporary certificates' duration is controlled by the **legacyCertificateValidity** value in the `/etc/opt/opsware/crypto/security.conf` file. The units are in days. Its suggested that at least 60 days be allowed between migration and receiving their unique 3rd party certificate. The process of how a managed server receives its 3rd party certificate is not part of the air-gapped migration document.

Tooling

The support tools are frequently used, and the documentation omitted the full paths for brevity's sake. Access to them and their manual pages is achieved by sourcing the setup file.

```
source /opt/opsware/support/setup.sh
```

Add this to a login profile, such as .profile or .bashrc, so they are always accessible.

Check KM000001427 for the latest update to date version. It's also crucial that the old mesh has the latest OPSWtools installed, as these are used to export data.

Building out the new environment

The official supplied documentation covers the details of building a new SA environment. Some items deserve mention before the migration occurs.

Cloning configuration

Optional. If the new environment is to run with the same tuning as the old environment, it can be an arduous task to locate and change every setting.
It noted that the wholesale copying of configuration parameters to a later SA version should be approached cautiously.
The **cora** tool can simplify this process;

OLD MESH: `/opt/opsware/support/bin/cora`
A file will be created. Copy this to the new mesh.
NEW MESH: `/opt/opsware/support/bin/cora -v restore old_mesh.cora`

To ensure all the .conf files are updated, the restore must be run on each slice of the new mesh.

SA 2022.11 ships with an improved baseline of component tuning parameters, and it might be prudent not to transfer the configuration from an older system if this SA version is the target.
An alternative approach is to DIFF the old environment with the new one and manually adjust those parameters required. **cora** can also assist with this.

```
cora -v html --diff prevmesh.cora newmesh.cora
```

At its most basic level, it requires a comparison of all the *_custom.conf settings and the database configuration tuning parameters.

Routing and firewalls

The managed servers in the old mesh must be able to contact their new mesh satellite using port 3001. This may necessitate new firewall and routing rules.

SA Patches

The latest rollup and agent bundle for the new environment should be installed. Note that the SA rollup patch may not install the latest OPSWtools, see the Tooling section of this document.

Available patches for Server Automation can be queried in the Support Portal.

LDAP integration

Integration with an external LDAP server can be set up now if used. If the exact configuration from the old mesh is used in the new mesh, the configuration can be easily copied with the following command instead of using the /opt/opsware/twist/ldap_config.sh tool.

```
OLD MESH:  /opt/opsware/support/bin/ldapAttrib -s > old_mesh_ldap_config.sh
NEW MESH:  sh ./old_mesh_ldap_config.sh
```

The User Groups created by the LDAP integration will not have any permissions assigned to them. This is taken care of in the data migration phase.

Once the configuration is on the new core, sync the users and groups by running.

```
NEW MESH: /opt/opsware/twist/ldap_sync.sh
```

Pluggable platforms

If pluggable platforms are used, these should be installed before migrating servers. All Operating Systems used by the old mesh must be present in the new mesh to enable the migration of servers and data.

These are available from the ITOM Marketplace

All core and satellite should be installed before the pluggable platforms are uploaded, as installing a pluggable platform will modify the SA infrastructure. Satellites and cores added after the platform has been installed will require the installer to be executed again.

Patch imports

Integration with external Vendor patch repository to import patches into SA should be set up to only import patches for the Operating systems that are still being maintained. For example, the old mesh may have Windows 2003 patches installed, and that platform is no longer required. As patches take up database and filesystem storage, the migration of the environment is the ideal time to jettison data that is no longer required.

The air-gapped migration tools can perform thin Patch Policy migrations reattaching those items imported into the new mesh.

Patch import configuration files may be copied from the old mesh to the new mesh

- /etc/opt/opsware/patch_importer/
- /etc/opt/opsware/rhn_import/
- /etc/opt/opsware/solpatch_import/
- /etc/opt/opsware/sles_import/

Windows patch import requires using a binary supplying parameters

- /opt/opsware/mm_wordbot/util/populate-opsware-update-library

Considerations: The scheduled execution of these jobs may be kicked off by a cron job and scripts on the old mesh

Audit compliance policies

If the Marketplace Connector (MpC) is used to import audit and compliance data, this must be set up and integrated into the new environment as a pre-requisite.

Instructions on how to accomplish this are provided as a practitioner's note

- https://docs.microfocus.com/doc/Server_Automation/2020.11/PN/pn601abc0e5c2aa6.69966527

Or via the official documentation for MpC

- https://marketplace.microfocus.com/itom/content/marketplace-connector-mpc

Depending on the SA version of the old mesh, the agents' age may be considered for the continued use of MpC, as later versions of the rule deliver python3 code for the rules, which requires a minimum agent version. This means that the agent must be upgraded during the server migration process before the audit policy will work again.

Prepare the old environment before migration

The existing SA environment must be in a healthy state before the migration of data and servers.

Removing unreachable servers

Given we will be moving all the servers from the old mesh to the new, servers that have been unreachable for an extended period must be cleaned up. This is important for several reasons.

If migrating from a pre-10.60 environment, there was no license check inside SA and no warning when capacity was exceeded. With the integration of AutoPass technology with SA 10.60 and beyond, the GUI will display various banners based on the installed license capacity. Unreachable servers have always cost money. Still, now it's visible to all SA users.

Exporting server relationship data for servers that have been unreachable for an extended period is unnecessary and adds to the export and import time during the data migration phase.

Once the server migration phase begins, you don't have time to investigate hundreds or thousands of unreachable servers and determine which should be migrated. This is when an SA outage occurs and should be kept to a minimum.

As of OPSWtools-2.0.7 a support tool called **unreachableServers** has been provided to assist with the testing, reporting, decommissioning, and deleting unreachable servers. From the manual tool page - consult for its full capabilities.

```
EXAMPLES
       Purge all servers greater than 720 days unreachable without a comms test.

       unreachableDevices --days 720 --delete --notest

       Report on servers unreachable for more than 180 days with no other action taken.

       unreachableDevices --days 180 --report
```

Python 3

When upgrading to SA 2020.11 and above, the agents uplift from Python 2 to Python 3, and any python code that is executing on the Agent must now be written in Python 3, or a python2/3 hybrid if its to remain in the old mesh, or it will fail when migrated.

The natural question is, "What users are executing Python scripts on the agents?"

The query is a little ghastly as the codetype is held as a parameter against a job in an XML markup syntax.

```
# sql "select d.username, a.session_id, session_desc
from sessions a, session_params b, session_param_values c, security_users d
where a.session_id = b.session_id
and b.session_param_name ='codetype'
and c.session_param_id = b.session_param_id
and text_data ='<value><string>PY2</string></value>'
and a.user_id = d.user_id"

Query #1 on Facility_id 1 (c1):
USERNAME | SESSION_ID | SESSION_DESC
-----------------------------------------------------
brett    | 18320001   | Run Server Script (Hello Python)
```

This will only locate Server Scripts. Code may also be embedded in Software Policies, OGFS Scripts, Custom Audit Rules and other locations inside SA. This must be uplifted to be Python3 compatible before migration so it continues to work in the new environment.

This may also necessitate that the old mesh managed server has a minimum Agent version with the **six** Python 3 compatibility module, so code made Python 3 compatible can continue to execute against a Python 2 agent.

Minimum agent version in the old mesh

- 2018.08 : Agents > internal build 80417
- SA 10.60 : Agents > internal build 80560
- SA 10.50 : Agents > internal build 81122
- SA 10.23 : Agents > internal build 81536

See also: https://community.microfocus.com/it_ops_mgt/dca/f/sws-sa_sup/1558/python-2-3-compatibility

Cleaning up Device Groups

This task involves removing data from the existing environment that does not need to be migrated. This general maintenance activity should be performed even when you are not migrating your mesh, as it helps improve overall performance. The OPSWtools provide tooling to help with this

- dgUsage - Identify device groups and their usage
- dgDump - View detailed information about dynamic group rules
- dgTree - list Device Groups in a tree-like format.

- cli - this general-purpose tool has delete, list, edit and tree actions that operate on device groups.

Operating System Supportability

When migrating between major SA versions, it's essential to understand that the higher version may have deprecated support for various OSs. An environmental assessment should be made to understand the number of servers associated with each OS and if those OSs are still in the support matrix for the target environment. If they are not, a decision must be made about what to do with them. It may be possible to migrate the existing agents to the new system. However, there may be some loss of functionality with this move. Also, note that later SA versions use Python 3 agent, and there may be compatibility issues using a Python 2 agent when the SA system expects a Python 3 agent.

Cleaning up the Software Repository

During the operational life of an SA environment, it will accumulate many software items in the Word repository, many of which are not required in the new environment. Deleting old software reduces the time it will take to perform a CBT export and import of the Software Repository.

The patch import processes may have imported software for Operating Systems no longer in use. These patches should be deleted from SA before the migration activities occur—for instance, Windows 2000.
The support tool **deletePlatformUnits** shipped with OPSWtools-2.0.13 can assist with this task. The purpose of this tool is to mark for deletion all the non-internal software associated with a Platform.

The Migration

The new mesh has been fully built, including installing satellites; the mesh has the latest rollup and agent bundle installed; the most recent OPSWtools released are installed; pluggable platforms have been installed along with patch and audit content, LDAP is integrated, and it's ready for the core of the air-gapped migration process.

At a high level, the two basic steps of moving data and moving servers, in reality, breaks down like this:

- pre-migration activities
- Server Migration and attach data (iterable)
    - Move servers from the old mesh to the new
    - post-server migration activities
- post-migration activities

The air-gapped migration process allows an iterative approach to move servers from the old mesh to the new mesh. This may be desirable if the system is divided into Customer tenants. All servers for a customer can be moved together, first, the servers are migrated, followed by server-related data.

There is a single iteration of the Server Migration step for the all-in-one-push approach.

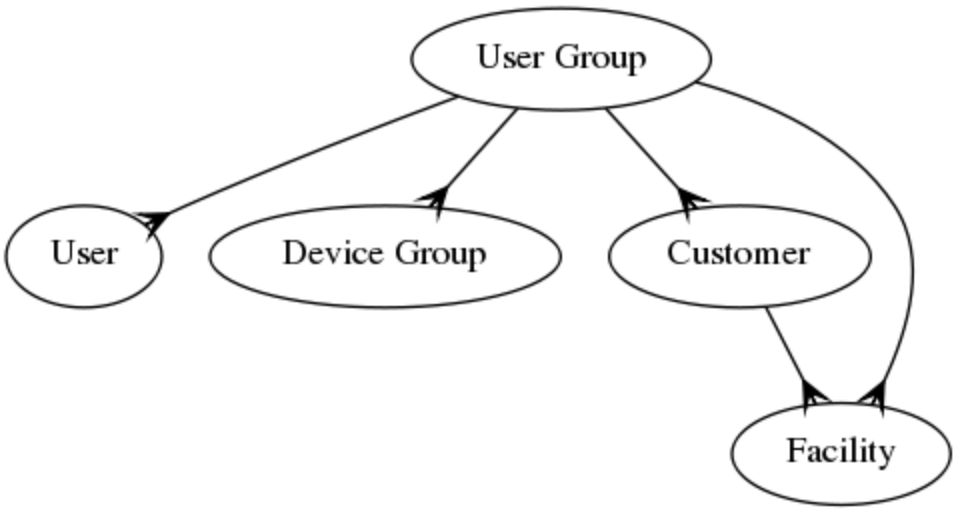
pre-migration activities

These task deal with the loading of data before we start swinging managed servers, aka Agents, from the old mesh to the new mesh.

The steps are as follows, in this order. The details of each step are expanded in the sections below.

- Migration of the Permission Model
    - Migrate Users
    - Migrate Customers and Customer/Facility linkages
    - Migrate Device Groups
    - Migrate User Groups and User linkages.
- Migration of OGFS home directories

Quite a bit of data needs to be moved before the User Groups can be imported. User Groups have relationships with Customers, Facilities, and Device Groups. They also have linkages to Users. All these objects must be present in the new mesh before a User Group can be imported from the old mesh.



This is then followed by the migration of

- Software Repository Library content
- Server relationships
- Recurring jobs
- Patches (optional)
- Patch Policies (optional)

Migration of the Permission Model

Migrate local users

Even if the SA system is LDAP integrated, the first task is to migrate local users. Often there are local and SA administration users, so if LDAP is not available, SA access is still available to those that require it. The user migration can be performed without assessment. It will be a NO-OP.

Assess if this step needs to be performed by executing these SQL statements on the old mesh.

```
# sql "select credential_store, count(*) from aaa.aaa_user where user_status='ACTIVE' group by credential_store"

Query #1 on Facility_id 1 (C1):
CREDENTIAL_STORE | COUNT(*)
--------------------------
TRUTH            | 11
DIRECTORY        | 55
```

The credential_store attribute is how SA identifies a user's authentication method. TRUTH is for local users, DIRECTORY is for LDAP users, and other credential stores exist.

There are several local users that SA itself has created. In this example, the **tom** user is our only local user. The **userMigrator** tool only exports non-SA-owned users.

```
# sql "select username from aaa.aaa_user where user_status='ACTIVE' and credential_store='TRUTH'"

Query #1 on Facility_id 1 (C1):
USERNAME
----------------------
admin
buildmgr
detuser
applianceserviceaccount
integration
agentuser
opswnobody
virt_scanner
v12nogfs
osprovuser
tom
```

Export the users from the old mesh

```
userMigrator -e old_mesh_users
```

Take the **old_mesh_users** file to the new mesh and import

```
userMigrator -i old_mesh_users
```

Migrate Customers

This step requires migrating Customers and their linkages to Facilities, which can be one of the most tedious steps.

Creation of customers. With the **cli** support tool, we can run this on the old mesh to produce a script we can copy and execute on the new mesh This requires the **jq** unix utility

```
yum -y install jq
```

These include the default customers Customer Independant, Not Assigned, Opsware, and Software Repository, which can be omitted when copying the output. In this case, only Mr Red is a migration candidate.

```
# cli customer list field=displayName,shortName output=json | \
jq -r 'map("cli customer add name=\""+.displayName+"\" shortName="+.shortName)|.[]'
```

The output is yet more **cli** commands to create these customers.

```
cli customer add name="Customer Independent" shortName=ANY
cli customer add name="Not Assigned" shortName=UNKNOWN
cli customer add name="Opsware" shortName=OPSWARE
cli customer add name="Software Repository" shortName=SOFTWARE_REPOSITORY
cli customer add name="Mr Red" shortName=MRRED
```

Unless the new mesh has precisely the same facility names creating the linkages will pose a challenge. A report to understand the relationships is easier than clicking around in the GUI.

```
cli customer report by=facility

Customer            |facility
--------------------|--------
Customer Independent|C1
Not Assigned        |C1, SATA
Opsware             |C1, SATA
Software Repository |C1
Mr Red              |C1, SATA
```

The report output can also be emitted in JSON for machine manipulation of the results, useful for **jq** transformations.

```
cli customer report by=facility output=json
```

Some helpful advice can be given here, as every mesh migration is different, and one solution may not fit all.

A **cli** script to build a **cli** script to re-create the Customer/Facility relationships. Executing on the old mesh we produce a script to copy to the new mesh. If the facility names have changed, this is a good starting point for editing and execution.

```
cli customer report by=facility output=json | \
jq -r 'map("cli customer +join name=\""+.Customer+"\" facility=\""+(.facility|sub(" ";""))+"\"")|join("\n")'
```

Output

```
cli customer +join name="Customer Independent" facility="C1"
cli customer +join name="Not Assigned" facility="C1,SATA"
cli customer +join name="Opsware" facility="C1,SATA"
cli customer +join name="Software Repository" facility="C1"
cli customer +join name="Mr Red" facility="C1,SATA"
```

To attach a customer to every available facility, we can generate a **cli** script by running this on the new mesh.

```
allcustomers=$(cli customer list output=json | jq -r 'map(.Customers)|join(",")')
for facility in $(cli facility list output=json | jq -r 'map(.Facilities)|join(" ")'); do
  echo cli facility +join name=$facility customer=\"$allcustomers\"
done
```

Example output (cut/paste to execute or remove the echo when generating).

```
cli facility +join name=C1 customer="Customer Independent,Not Assigned,Opsware,Software Repository,Mr Red"
cli facility +join name=SATA customer="Customer Independent,Not Assigned,Opsware,Software Repository,Mr Red"
```

Sample execution of a single line

```
# cli facility +join name=C1 customer="Customer Independent,Not Assigned,Opsware,Software Repository,Mr Red"
INFO : Adding Customer ['Customer Independent', 'Not Assigned', 'Opsware', 'Software Repository', 'Mr Red'] to
Facility 'C1'
```

You will find that **cli** and **jq** are your friends for this data migration step; these tools are also handy for general administration when adding a new customer to the system and attaching it to multiple user groups and facilities.

Migrate device groups

The migration of the Device Groups can be performed with the **migrate** tool.

```
OLD MESH:  migrate --device_group all -e old_mesh_device_groups
NEW MESH:  migrate -i old_mesh_device_groups
```

Some useful commands if the configuration of the old mesh is not the same as the new mesh and additional Customers or Facilities need to be added to the User Groups.

Join every User Group with Cust A and Cust B

```
while read -r gname; do
    cli usergroup +join name="$gname" customer="Cust A,Cust B"
done < <(cli usergroup list | tail -n+3)
```

Add a facility to every user group

```
allgroups=$(cli usergroup list output=json | jq -r 'map(.UserGroups)|join(",")')
cli facility +join name=FAC "usergroup=$allgroups"
```

An alternative implementation that iteratively calls **cli** (slightly slower).

```
while read -r gname; do
    cli usergroup +join name="$gname" facility="FAC"
done < <(cli usergroup list | tail -n+3)
```

The **migrate** tool has a couple of limitations.

- Device Groups with the / character in their name cannot be migrated.
- It does not perform topological sorting, only hierarchical ordering (parents before children). This means another device group may refer to a Device Group before it has been created. Multiple import runs are required to overcome this problem.

The CBT tool can also be used. However, this introduces another problem CBT will export the Device Group and all referents. This creates a much larger export/import than using **migrate** it may also bring across data you want to discard. It's for this reason that the **migrate** program is preferred.

servergroup.rdf

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY filter "http://www.opsware.com/ns/cbt/0.1/filter#">
]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://www.opsware.com/ns/cbt/0.1/filter#">
  <ServerGroupFilter rdf:ID="sg1">
    <path>/Group/Public</path>
    <directive rdf:resource="&filter;Descendants"/>
  </ServerGroupFilter>
</rdf:RDF>
```

Execution will output into the *dg-cbt/* folder.

```
/opt/opsware/cbt/bin/cbt -e dg-cbt -f servergroup.rdf
```

Take the *dg-cbt/* folder to the new mesh and import

```
/opt/opsware/cbt/bin/cbt -i dg-cbt
```

User Group Migration

With Customers, Facility and Device Groups in place, the User Groups can be migrated.

There are some considerations for the export. If the mesh's topology has changed and the facilities in the new mesh don't map 1:1 with those from the old mesh, you may wish to skip exporting the User Group/Facility mappings. Alternatively, it can be exported and transformed into new facility names on import using the -t translate option.

This tool will be involved in the process, but there is no turn-key solution as it depends on each migration. See the tools manual page for more details.

Exporting on the old mesh_ taking everything

```
userGroupMigrator -e user_groups_old_mesh
```

Exporting on the old mesh_ but not the facility or folder relationships

```
userGroupMigrator -e user_groups_old_mesh --skip_facilities --skip-folders
```

The import on the new mesh

```
userGroupMigrator -i user_groups_old_mesh
```

Migration of OGFS home directories

The user's OGFS home directory is their Global Shell home directory. This directory will exist on each core, which may contain different content as it's not synchronized across the mesh.

It's not mandatory to copy over all the user's OGFS home directory; if needed, there are some considerations.

Their folder will reside on the filesystem of Slice0 for each core, and its NFS mounted to the slices. An example path; /var/opt/opsware/ogfs/export/store /home/brett The entire directory /var/opt/opsware/ogfs/export/store/home can be copied from the old mesh to the new mesh to keep the user's content.

The problem that needs to be solved is that when the users are created, either via LDAP integration or the **userMigrator** program in the new mesh, they will be given different OGFS Hub uid and Hub gid numbers, and these will be out of sync with values from the old mesh. This will cause a filesystem permission problem when the users try to access their copied filesystem content.

This problem is solved by executing the support tool APX utilities.resyncOGFSuserFiles on the new mesh after the files have been copied.

```
# /opt/opsware/bin/apxtool import /opt/opsware/support/APX/utilities.resyncOGFSuserFiles
APX with unique name 'utilities.resyncOGFSuserFiles' does not exist.
Register it into the system? Y/N y
Info: Successfully registered APX 'Resync orphaned OGFS user files' (4210001) in folder '/Opsware/Tools
/Administrative Extensions'.
Info: Successfully published a new version '1' for APX 'Resync orphaned OGFS user files'.
Info: Successfully set APX 'Resync orphaned OGFS user files'(4210001) current version as '1'.
#
```

Locate the APX using the GUI and execute it.

Software Repository Library content

Each top-level folder with customer-specific content should be listed for export.
Review the file */opt/opsware/cbt/filters/folder.rdf* for more examples of constructing the export filter.

library.rdf

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY filter "http://www.opsware.com/ns/cbt/0.1/filter#">
]>

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns="http://www.opsware.com/ns/cbt/0.1/filter#">

<FolderFilter rdf:ID="f1">
  <path>/Customer-Folder1</path>
  <recursive rdf:resource="&filter;Yes" />
</FolderFilter>

<FolderFilter rdf:ID="f2">
  <path>/Customer-Folder2</path>
  <recursive rdf:resource="&filter;Yes" />
</FolderFilter>

</rdf:RDF>
```

Execution will output into the *library-cbt/* folder.

```
/opt/opsware/cbt/bin/cbt -e library-cbt -f library.rdf
```

Take the *library-cbt/* folder to the new mesh and import

```
/opt/opsware/cbt/bin/cbt -i library-cbt
```

Exporting and importing library content can take a significant amount of time.

Server relationships

Before servers are migrated to the new mesh, their relationship data must be exported. The **migrate** program handles this relationship data.

- Audits
- Audit Exceptions
- Device groups
- Snapshot Specifications
- Software Policy
- Patch Policy
- Device History, Customer, Use, Stage and Custom Attributes

Export all data. Even if migrating servers iteratively, you should take a full export.

```
migrate --all -e old_mesh_server_relships
```

You must export the data before migration to the new mesh, as this data will not be available once the server has been attached to the new mesh and decommissioned from the old mesh.

The `old_mesh_server_relships` data cannot be used until the server has swung to the new mesh. Surrogate keys for the server will be used to locate the data and reattach it. Why surrogates? The Unique Primary Keys change when data is migrated to the new mesh. For instance, a server will obtain a new MID depending on the order in which it was migrated. The migration program will attempt to locate the same server using its UUID and HOSTNAME.

For this reason, before the export is run, duplicate servers must be corrected with either a hostname or a UUID change. They can be located with this SQL executed on the old mesh.

```
select system_name, uuid, count(*)
from devices
where opsw_lifecycle='MANAGED'
group by system_name, uuid
having count(*) > 2
```

Correct to have a unique FQDN before migrating. It's not uncommon to have multiple servers with the name **localhost**.

Recurring jobs

On the old mesh recurring jobs should be exported. The import should only be run once ALL of the servers have been swung over to the new mesh

```
migrate -e old_mesh_jobs --jobs
```

Server Migration

This phase swings the server from the old mesh and attaches them to the new mesh, decommissioning the server from the old mesh upon successful migration. The server attachment data must have been exported before migration, as it will be lost during the decommissioning phase.

The support tool used for this phase is an APX, and it must be imported into the old mesh with the bootstrap certificate from the new mesh to allow the server to register. More detail about this tool can be found on its manual page; /opt/opsware/support/APX/com.hp.device.migrator/README.txt

In migrating with the all-in-push approach

- Execute the APX on all the servers in the old mesh to move them to the new mesh
- on the new mesh # migrate -i old_mesh_server_relships

This step can be iterated by migrating servers in waves

- Execute the APX on a group of servers in the old mesh
- on the new mesh `# migrate -i old_mesh_server_relships --device_list server_list_file`

If using a device group to maintain the servers to migrate, the file server_list_file can be created like this

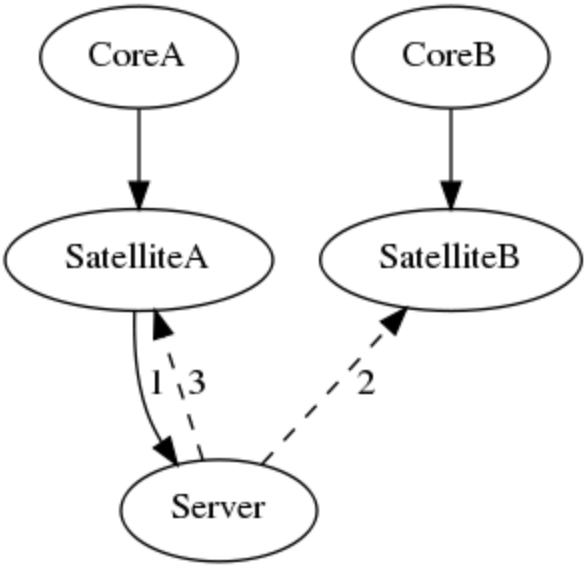
```
cli devicegroup list path=/migration_wave output=csv | tail -n+2 > server_list_file
```

The servers should be removed from this device group once decommissioned so they don't appear in the `server_list_file` for the next wave.

Agent migration workflow

Understanding the APX migration architecture allows better reasoning about how and why it failed. At a high level:

1. The APX reaches down to the server and pushes some migration logic, and this is executed asynchronously, in batch mode, disconnected from the installed Agent.
2. Migrate code puts the server back into bootstrap mode and points the Agent at the new GW, obtaining *new crypto* and starting the agent. The original agent crypto and gateway should be restored if the operation fails.
3. The migration logic connects to the server's original GW using *old crypto* and updates the core with a status. The APX is waiting to see this success or failure status for server job status reporting.



The status value is hidden custom attribute `__OPSW_device_migrator` stored against the server.

post-migration activities

Restoring recurring jobs

The import should only be run once ALL of the servers have been swung over to the new mesh

```
migrate -i old_mesh_jobs
```

Importing jobs is not idempotent, and multiple import executions will create multiple recurring jobs.

Agent Upgrades

The APX only migrates the agent. It does not perform any upgrade activity. All agents must be upgraded.

All OS build processes that have an agent baked into them must be adjusted.

If moving from a pre-10.60 SA version to a post-10.60 version, the manual agent installation process has changed. This may require an overhaul of your manual agent installation processes.

- https://docs.microfocus.com/doc/Server_Automation/2020.11/InstallSAAgentCLI

Tuning

It's recommended to run the following support tools to perform automated tuning.

```
/opt/opsware/support/bin/lessSpinErrors
/opt/opsware/support/bin/tuneSA
```

The usage of tuneSA will disable functionality that is not currently used.
Read the manual page to understand what will be affected.

Monitoring

The primary slice of each core maintains a cache of 100 certificates. These are distributed to new servers as they are registered. If you use a 4096-bit key size, the cache will refill slower than a 2048-bit key. This can be problematic if you schedule 1000 servers to move in a single batch, as the cache may empty, causing agents to wait.

As a baseline, 250 certs with a 4096-bit key size take about 1m30s to generate. The default key size is 2048 bits, and 100 certificates build in 15s. The agent will wait to 2m for a certificate before timing out. With the default key size, it's harder to overrun the cache. Check the keysize to determine if cache tuning is required. Consult `/opt/opsware/support/APX/com.hp.device.migrator/README.txt` for tuning instructions.

/etc/opt/opsware/crypto/security.conf

```
certificateMode: legacy
fips: 0
keysize: 2048
md: sha256
```

If you wish, monitor the new mesh certificate caches during the migration when scheduling a massive batch of servers.

You can see how many are in the cache like this

```
# ls /var/opt/opsware/spin/cadb/realm/opsware/certs | wc -l
    100
```

The certificate generation was sped up significantly with patch OCTCR19L1456410, appearing in SA 2020.11.005, and SA 2022.11.001, and helps mitigate cert cache exhaustion.