**Continuous Integration/Deployment –CA API GATEWAY**

**API M Practice**

Miracle Software Systems, Inc.

Steps:

1. Develop a policy in policy manager
2. **Export a Policy:**Export the policy xml using headless java utility called Gateway Migration Utility, available from CA, and check in the artifacts
   1. Create an argFile.properties with below arguments

host – host name of the gateway from which the policy has to be exported

port – Port number to connect to the gateway

username – username to connect to the gateway

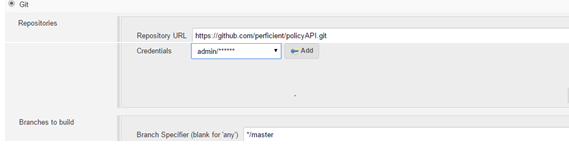
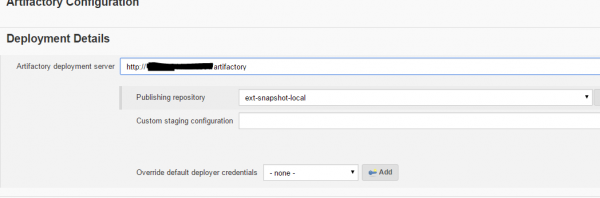
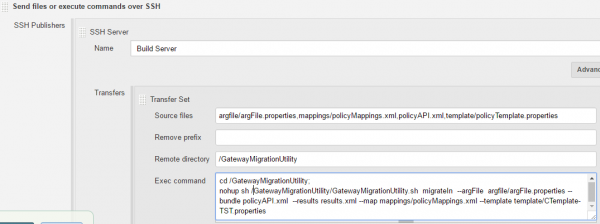
password – encoded password of the gateway

format – specify as single file if the export is for policy file or folder for a folder

dest= policy XML name for storing the XML

serviceName= Actual Policy name in the gateway

encryptionPassphrase – Passphrase for encryption

1. Before export , validate the connection to the gateway is successful using gateway migration utility command below
   1. GatewayMigrationUtility.sh browse -z argFile.properties -r –showIds – this should list all the policies running in the gateway.
2. Migrate out the policy from the gateway by using the gateway utility command below
   1. GatewayMigrationUtility.sh migrateOut –argFile argFile.properties
3. Externalize the variable with the values from the exported policy xml using the below command
   1. GatewayMigrationUtility.sh template –bundle policyAPI.xml –template policyTemplate-.properties – This will create the template properties file with the key / value pair having variable names and corresponding values, and the values in the file can be modified with specific environment properties.
4. Create a Mapping file for the mappings. These are used to specify the action to be used for a given type and the srcId when importing to different environments.
   1. Copy the mappings section from the bundle file
   2. Save the mappings as policyAPIMappings.xml
   3. Modify the properties for the type as required
5. Check in the policy XML, Mappings XML, argFile properties file, and template property files to the code repository
6. **Import a Policy:**Configure Jenkins to poll the files for the policy from code repository[](http://blogs.perficient.com/integrate/files/2016/05/Jenkins1.png)
7. Package the artifacts (policy XML , mappings XML , argFile and template properties file ) to artifactory or any build file storage[](http://blogs.perficient.com/integrate/files/2016/05/Jenkins2.png)
8. Configure  SSH to send the files over to build server and execute the migrate in command as below[](http://blogs.perficient.com/integrate/files/2016/05/Jenkins3.png)
9. Configure the test cases for the policy as an test suite to execute the test cases after the migration to the target gateway and publish the results

Note: The above example uses credentials (username/password) of the gateway to import and export the policy out of gateway. You can do this using the certs as a mechanism to import /export the policy.

**GMU Walkthrough**

This walkthrough/tutorial describes how to use the GMU to migrate policy across Gateway clusters.

It borrows heavily from and builds on the good work of Derek Orr, who created much of the original PPT deck. If you are comfortable with the Gateway and Policy Manager you may find the updated PPT more useful, as it omits many of the basic steps described below, although the end result is the same.

You will:

1. Create test services on your source Gateway that include Cluster-Wide Properties
2. Create a new admin user ‘restman’ on the source and target Gateways
3. Publish the Gateway REST Management Service (/restman) on both the source and target Gateways
4. SFTP the GMU files to the source Gateway
5. Create and update directories and files to simplify the GMU file migration
6. Use the GMU to migrate the folder of services from the source to target Gateway
7. Use the GMU to migrate a single service from the source to target Gateway

You need:

* Source Gateway in v9.0 or above; in this example the source Gateway is named test1.90.ca
* Target Gateway (version must be identical to source Gateway); in this example the target Gateway is test3.90.ca
* Updated /etc/hosts file on both source and target gateways so that they know each other’s ip address and hostname
* Policy Manager to connect to the source and target Gateway
* Latest GMU zip file
* Filezilla or similar FTP client
* Optional, but useful: Putty or similar ssh client
* Optional: Linux server to run the GMU commands; we will use the source Gateway as our server
* Link to GMU documentation included with GW v9.1: <https://docops.ca.com/ca-api-gateway/9-1/en/upgrade-migrate-patch-back-up-restore/migrate-gateways>

# Shortcuts

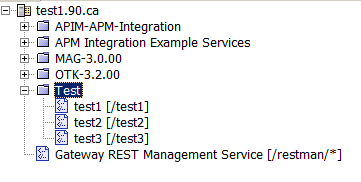
* [Start from scratch](#_Create_Example_Gateway) - let’s build some example services.
* [Already have services, just need to get started on a migration](#_Add_a_New).
* [Is it Friday yet?](http://isitfridayyet.net/)

# Create Example Gateway Services that Include Cluster-Wide Properties

**Skip this section if you already have sample services on your Gateway that you want to migrate. Proceed to** [**Add a New User Named ‘restman’ on the Source and Target Gateway with Administrator Permissions**](#_Add_a_New_1)

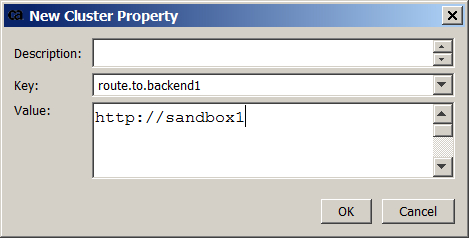
Connect to the Source Gateway with the Policy Manager.

Create the test services you plan to migrate. In this example we will use test1, test2, and test3, as shown below.

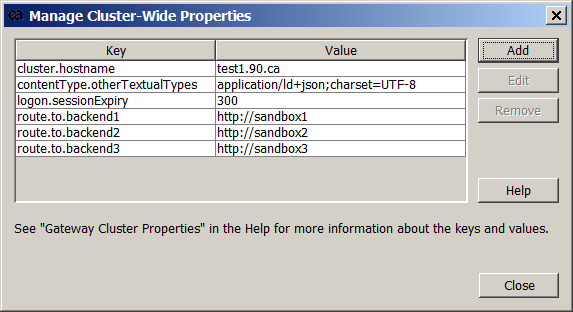


Add some placeholder policy to your services. In the example policy we use a proxy that uses Cluster-Wide Properties to route to backend <http://sandbox> on test1. The same policy will route to a different location (<http://qa>) on the target cluster. These values will be unaffected by the migration; they serve to demonstrate how we support cluster-specific variables.

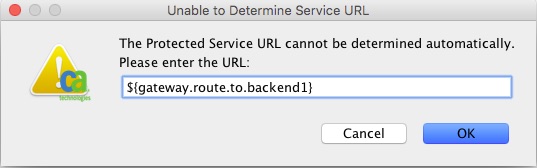
Create a Cluster-Wide Property in the source Gateway and set the values as shown in the figure below.



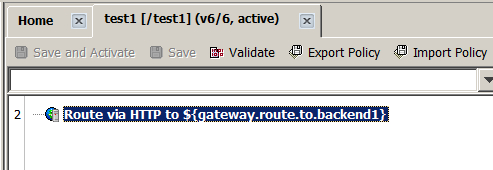
Add two more Cluster-Wide Properties for different backend locations, as shown in the figure below.

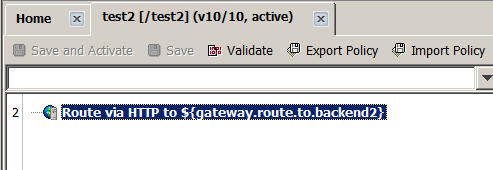


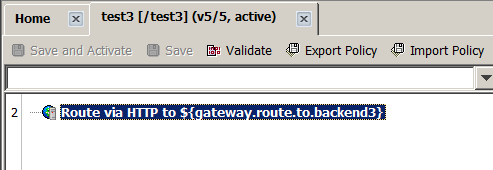
From there you can use the new Cluster-Wide Property in Policy. To do so, from the Message Routing list, click and drag Route via HTTP(S) into the service. Update the assertion so that it uses the new Cluster-Wide Property to route, as shown in the figure below.



Click OK to continue, and repeat the steps to add similar policy to /test2 and /test3, as shown in the figures below.







The example services are ready.

# Add a New User Named ‘restman’ on the Source and Target Gateway with Administrator Permissions

Connect to the source Gateway with Policy Manager.

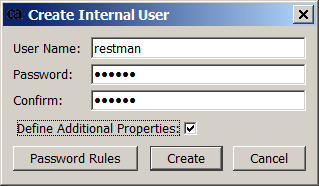
This example uses a password that is not allowed by default; you must update your password settings (right-click **Internal Identity Provider** and choose **Manage Password Policy**) if you wish to use it.

Right-click **Internal Identity Provider** and choose **Create User**.

Enter the values as described below:

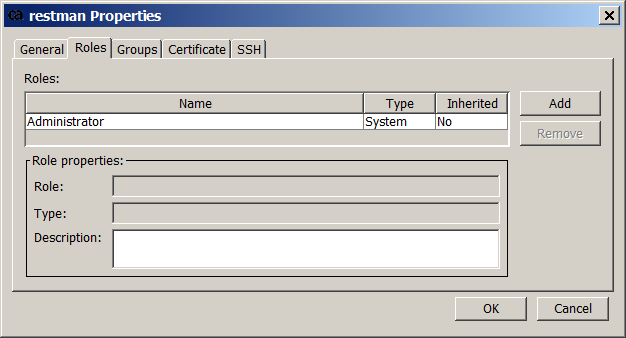
* **User Name**: restman
* **Password**: 7layer
* **Confirm**: 7layer

Click the **Define Additional Properties** checkbox.



Click **OK** to continue.

Set the Properties for restman and give the user the **Administrator** role, as shown in the figure below.



Click **OK** to continue. The user restman has been added as an admin on the source Gateway.

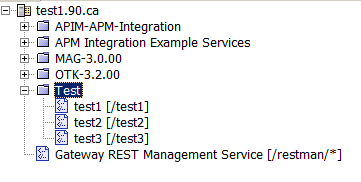
**Repeat the steps on the target Gateway using the same User Name and Password as you used on the Source Gateway.** To migrate policy you must have the restman user on both the source and target gateways, and the user must have the Administrator role on both gateways.

# Publish Gateway REST Management Service (/restman)

Connect to the source Gateway via the Policy Manager.

From the Tasks menu choose Publish Internal Service.

In the Service to Publish field select Gateway REST Management Service. Leave the default values and click OK. restman is listed with your other services, as shown in the figure below.

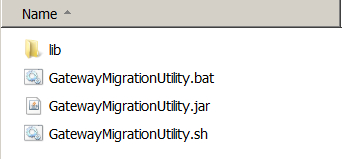


**Repeat the steps on the target gateway to add restman there.**

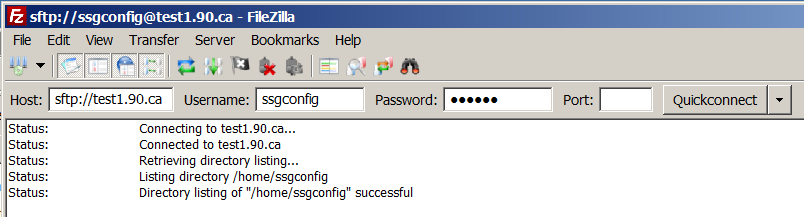
# SFTP the GMU Files to the Source Gateway

In an actual deployment you would add the GMU to a server and run it from there. In this example it is easier not to spin up another server so we will run the commands from the source Gateway. To do so we must add the files first.

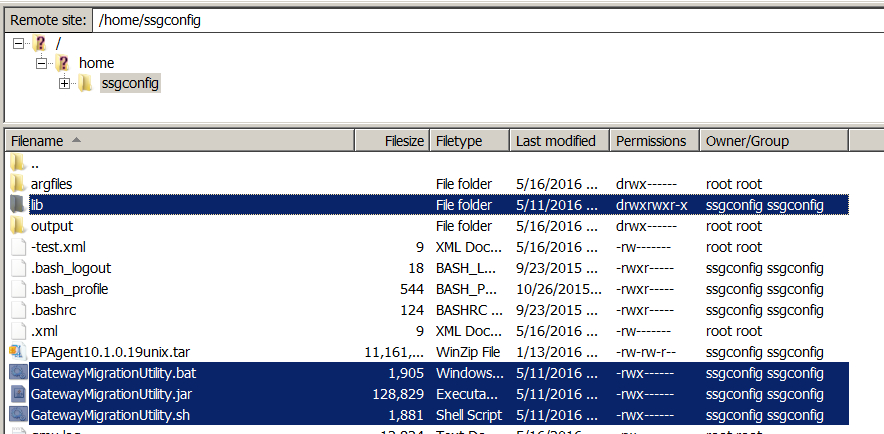
Unzip the GMU files to a local directory. In this example we use GMU v1.1. The files are shown in the figure below.



Use Filezilla or similar to connect to /home/ssgconfig on the source Gateway, as shown below (password in this example is 7layer).



Copy the files to /home/ssgconfig on the source Gateway, as shown in the figure below.



# Create and Update Directories and Files to Simplify the Migration

In order for this example to work you must update the /etc/hosts file on both the source and target gateways:

* Update the /etc/hosts file on the SOURCE gateway with the ip address / hostname for the TARGET gateway.
* Update the /etc/hosts file on the TARGET gateway with the ip address / hostname for the SOURCE gateway.

Using the GMU is easier if you create directories and files for your arguments (**argfiles**) and output (**output**). We will reference these files and directories when we run the GMU commands.

* The argument files directory (**argfiles**) stores the **source.args** and **destination.args** files. The args files specify the information you need to connect to the source and destination (target) gateways.
* The **output** directory is a destination for the XML output created when you use the GMU to run commands like migrateOut, migrateIn --test, and migrateIn.

This example migration is the least secure and not appropriate for a customer environment; customers will want to use the flow [described in the GMU docs](https://docops.ca.com/ca-api-gateway/9-1/en/upgrade-migrate-patch-back-up-restore/migrate-gateways/configure-gmu-and-gateways-for-migration).

**To create the directories:**

Log into the Gateway via a terminal or SSH via Putty.

Open a privileged shell (root).

**Navigate to /home/ssgconfig and create the directories:**

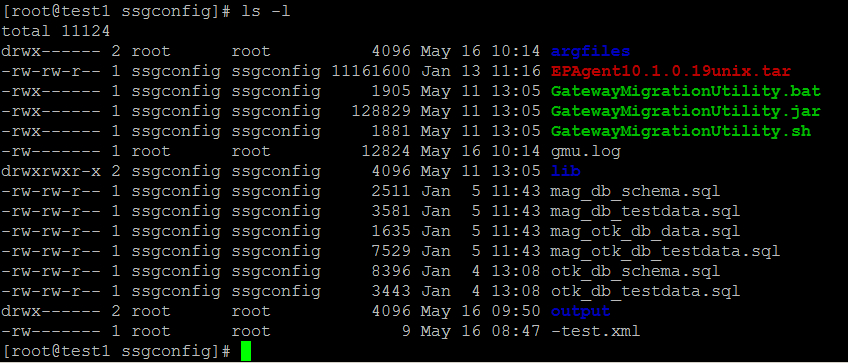
#cd /home/ssgconfig

#mkdir output

#mkdir argfiles

Confirm that you created the directories successfully:

#ls –l



**Create GMU argument files for the source and destination hosts.**

Note that the username and password in these files matches the User Name and Password we used to create the new admin ‘restman’ on both the source and target gateways.

The **source.args** file includes credentials for our restman user and sets trust values for the hostname and certificate.

**To create the source.args file:**

#cd /home/ssgconfig/argfiles

#vi source.args

Enter the following values in source.args:

host=\*\*\* Your SOURCE hostname \*\*\*

username=restman

plaintextPassword=7layer

port=8443

trustHostname

trustCertificate

The plaintext password is not best practice; use an encoded password in your environment ([as described in the GMU documentation](https://docops.ca.com/ca-api-gateway/9-1/en/upgrade-migrate-patch-back-up-restore/migrate-gateways/configure-gmu-and-gateways-for-migration)).

The **destination.args** file includes credentials for our restman user and sets values trust values for the hostname and certificate.

**To create the destination.args file:**

#cd /home/ssgconfig/argfiles

#vi destination.args

Enter the following values in destination.args:

host=\*\*\* Your DESTINATION hostname \*\*\*

username=restman

plaintextPassword=7layer

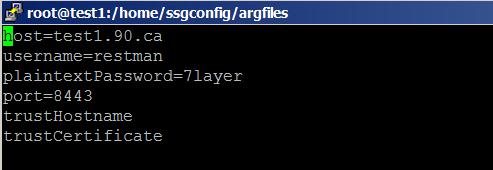
port=8443

trustHostname

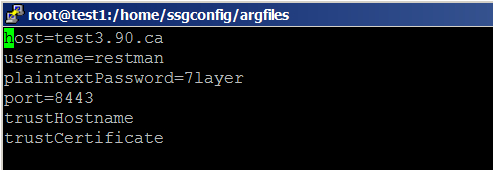
trustCertificate

As before: the plaintext password is not best practice; use an encoded password in your environment ([as described in the GMU documentation](https://docops.ca.com/ca-api-gateway/9-1/en/upgrade-migrate-patch-back-up-restore/migrate-gateways/configure-gmu-and-gateways-for-migration)).

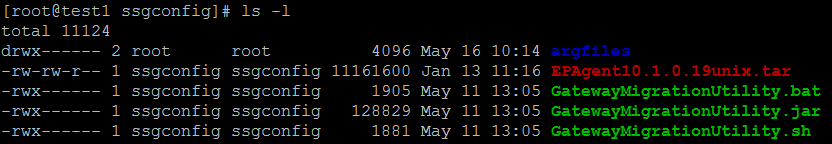
When complete your source.args file looks like this:



And your destination.args file looks like this:



Confirm that you have rwx permission on the GatewayMigrationUtility\* files, as shown in the figure below.



If you do not have rwx permission then update the permissions:

#cd /home/ssgconfig

#chmod 700 Gate\*

You will need an encoded password file.

**Run the following command in the /ssgconfig directory to encode your password:**

[root@test1 ssgconfig]# ./GatewayMigrationUtility.sh encodePassword

Enter the password to encode:

<enter your password>

Running.

FD271H0sGZA.Hb5oSho2jWfS69zUdg32og

**Copy the encrypted value into a new password file:**

[root@test1 ssgconfig]# vi argfiles/password

FD271H0sGZA.Hb5oSho2jWfS69zUdg32og

We will reference these files when we run the GMU.

Now that the files and directories are prepped we can run a migration.

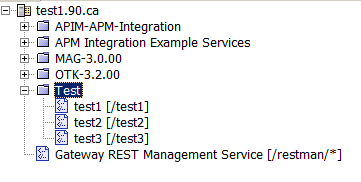
# Use the GMU to Migrate a DIRECTORY (Folder) of Services from the Source to Target Gateway

In order, we will:

1. View the list of GMU commands.
2. Browse the list of services to find the directory we want to migrate.
3. migrateOut the folder contents to a directory.
4. migrateIn --test to check for any conflicts
5. migrateIn
6. Check the policy on the updated Gateway

The example assumes that you are using the files and directories prepped in the sections above.

In the example the source Gateway is test1.90.ca, and the source directory with the test policies is called Test, as shown in the figure below.

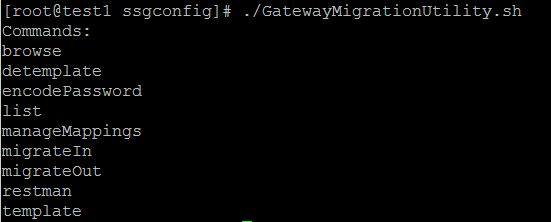


**View the list of GMU commands.** To do so, navigate to the location of the GatewayMigrationUtility.sh and run it from the command line.

#cd /home/ssgconfig

#./GatewayMigrationUtility.sh

The commands are shown in the figure below.



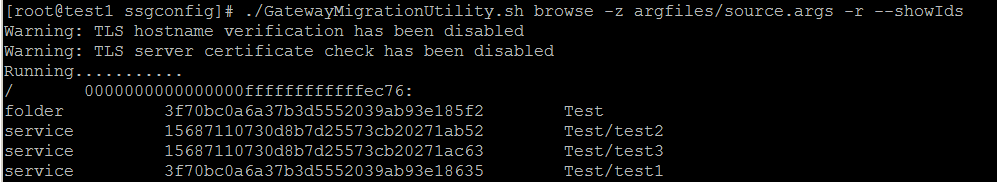
**To browse the services on the Gateway, type:**

#./GatewayMigrationUtility.sh browse -z argfiles/source.args -r --showIds

Where:

* -z identifies a file name for passing common GMU arguments. In this example we use the argfiles/source.args file to connect and provide credentials to the Gateway.
* -r shows a recursive list of all services; not strictly required for this example
* -- showIds lists the unique IDs assigned to the services by the Gateway; not strictly required for this example

Determine the path and filename of the service you want to migrate. In this example we will use Test, which is at the top level. Migrating /Test will scoop up all the services inside that folder (see figure below).



**Run migrateOut to output the file to your output directory:**

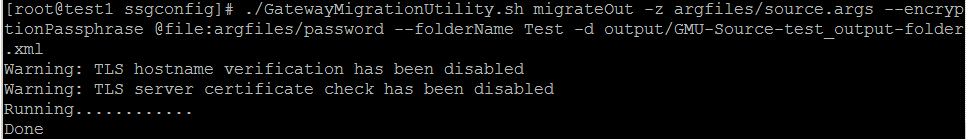
#./GatewayMigrationUtility.sh migrateOut -z argfiles/source.args --encryptionPassphrase @file:argfiles/password --folderName **Test** -d output/**GMU-Source-test\_output-folder.xml**

Bold text in the command above indicates arguments that you can change to match your own environment.

Where:

* -z identifies a file name for passing common GMU arguments. In this example we use the argfiles/source.args file to connect and provide credentials to the Gateway
* --encryptionPassphrase uses the password saved in the file in argfiles/password
* --folderName is the name of the folder you want to migrate
* -d sets the destination of the output

Note that if you have spaces in your folder name you must use escaped characters in the above command. It is easier to [build migration friendly policies](https://docops.ca.com/ca-api-gateway/9-1/en/upgrade-migrate-patch-back-up-restore/migrate-gateways/prepare-for-migrations/develop-migration-friendly-policies) without spaces.



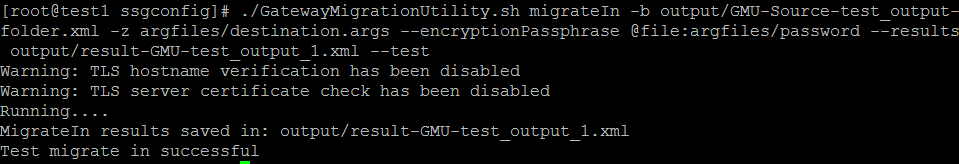
**Run migrateIn --test to check for conflicts:**

#./GatewayMigrationUtility.sh migrateIn -b output/**GMU-Source-test\_output-folder.xml** -z argfiles/destination.args --encryptionPassphrase @file:argfiles/password --results output/**result-GMU-test\_output\_1.xml** --test

Bold text in the command above indicates arguments that you can change to match your own environment.

Where:

* -b identifies the bundle you want to test migrate
* -z identifies a file name for passing common GMU arguments. In this example we use the argfiles/source.args file to connect and provide credentials to the Gateway
* --encryptionPassphrase uses the password saved in the file in argfiles/password
* --results identifies the location and filename of the test results
* --test sets this migrateIn as a basic import check

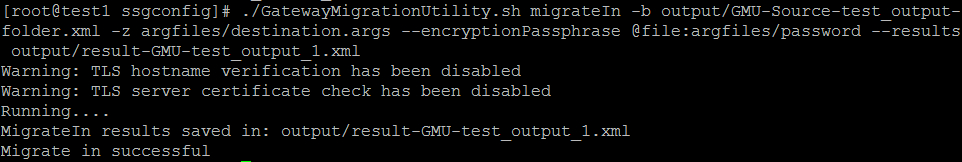


**Run migrateIn to import the folder to the target gateway:**

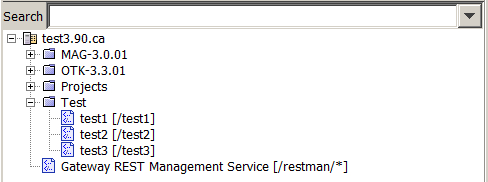
#./GatewayMigrationUtility.sh migrateIn -b output/GMU-Source-test\_output-folder.xml -z argfiles/destination.args --encryptionPassphrase @file:argfiles/password --results output/**result-GMU-test\_output\_1.xml**

Where:

* -b identifies the bundle you want to migrate
* -z identifies a file name for passing common GMU arguments. In this example we use the argfiles/source.args file to connect and provide credentials to the Gateway
* --encryptionPassphrase uses the password saved in the file in argfiles/password
* --results identifies the location and filename of the results of your migration



Use Policy Manager to connect to the target Gateway and confirm the migration, as shown below.

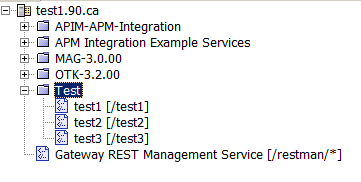


# Use the GMU to Migrate a SINGLE SERVICE from the Source to Target Gateway

In order, we will:

1. View the list of GMU commands that shows the unique ID for each folder and service.
2. Browse the list of services to find the service we want to migrate.
3. migrateOut the service contents to a directory.
4. Manage the mapping so that the updated service is added to the correct location without conflicts
5. migrateIn --test to check for any conflicts
6. migrateIn
7. Check the policy on the updated Gateway

In the example the source Gateway is test1.90.ca, and the source service is test2, as shown in the figure below.



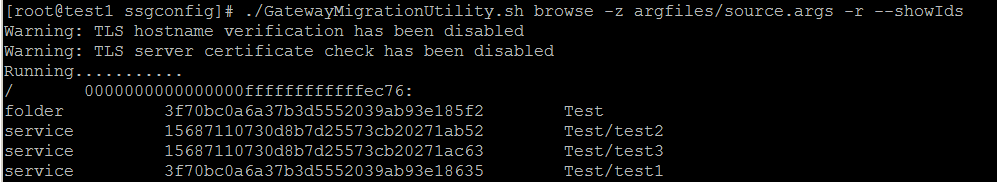
Update the test2 policy - any update you can recognize after you run the migration will work. Save and Activate the policy after you update.

**Use the browse command with the showIds argument to find the ID of your service:**

#./GatewayMigrationUtility.sh browse -z argfiles/source.args -r --showIds

Where:

* -z identifies a file name for passing common GMU arguments. In this example we use the argfiles/source.args file to connect and provide credentials to the Gateway.
* -r shows a recursive list of all services; not strictly required for this example
* -- showIds lists the unique IDs assigned to the services by the Gateway; not strictly required for this example



Note that the ID for /test2 is 15687110730d8b7d25573cb20271ab52. We will use this value in the migrateOut command.

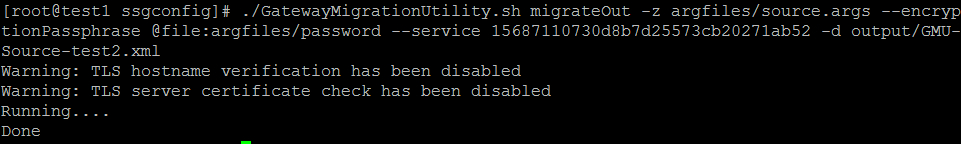
**Run migrateOut to output the service as a file to your output directory:**

#./GatewayMigrationUtility.sh migrateOut -z argfiles/source.args --encryptionPassphrase @file:argfiles/password --service **15687110730d8b7d25573cb20271ab52** -d output/**GMU-Source-test2.xml**

Bold text indicates values you should change for your own example.

Where:

* -z identifies a file name for passing common GMU arguments. In this example we use the argfiles/source.args file to connect and provide credentials to the Gateway
* --encryptionPassphrase uses the password saved in the file in argfiles/password
* --service is the ID of the service (or folder) you want to migrate
* -d sets the destination of the output file



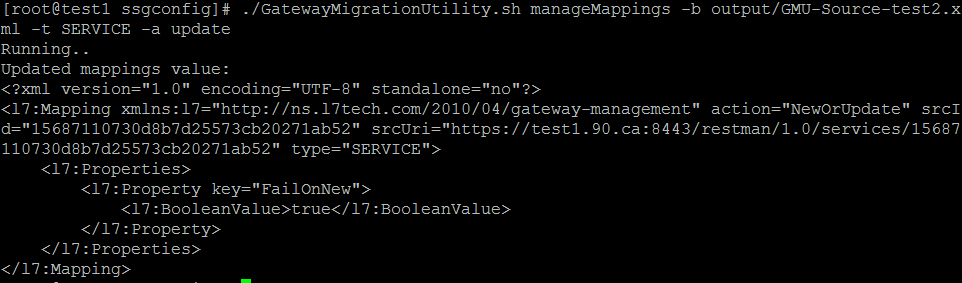
**Run manageMappings to manage conflicts:**

#./GatewayMigrationUtility.sh manageMappings -b output/**GMU-Source-test2.xml** -t SERVICE -a update

Bold text indicates values you may want to change to match your own example.

Where:

* -b identifies the bundle you want to update
* -t sets the type of edit you are making; for example: SERVICE, JDBC\_CONNECTION, ASSERTION\_ACCESS, CASSANDRA\_CONFIGURATION, etc.
* -a specifies the mapping action - in this example we use update, which will update the existing entity and fail if the entity does not exist. See the GMU manageMappings doc for a [list of the GMU mapping actions](https://docops.ca.com/ca-api-gateway/9-1/en/upgrade-migrate-patch-back-up-restore/migrate-gateways/gmu-command-help/managemappings-command)



With migrateOut and manageMappings successful you can run migrateIn to update your service.

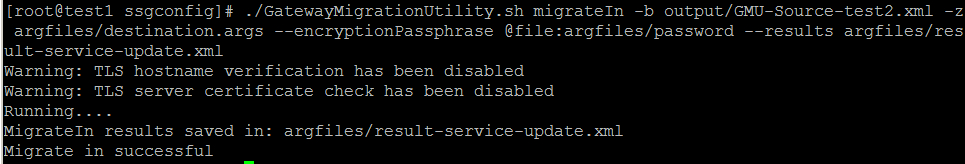
**Run migrateIn to update your service:**

#./GatewayMigrationUtility.sh migrateIn -b output/**GMU-Source-test2.xml** -z argfiles/destination.args --encryptionPassphrase @file:argfiles/password --results argfiles/**result-service-update.xml**

Bold text indicates values you may want to change to match your own example.

Where:

* -b identifies the bundle you want to migrate
* -z identifies a file name for passing common GMU arguments. In this example we use the argfiles/source.args file to connect and provide credentials to the Gateway
* --encryptionPassphrase uses the password saved in the file in argfiles/password
* --results identifies the location and filename for the results of your migration



Log into the target Gateway and confirm that the policy has been updated as expected.