

Create upgrade rules for: File Based Config

One requirement after making a schema change is to write an FBC upgrade rule to perform commit-to-commit upgrade within [master.groovy](#). All FBC upgrade is manual via the use of the [openam-config-upgrader](#) and its rule files. The upgrader's associated rules perform that transformation. This page is here to help you get started with writing FBC upgrade rules for any schema change.

- [tl;dr](#)
- [FBC files and their purpose](#)
 - [Description](#)
 - [Structure](#)
- [Rules to upgrade FBC](#)
 - [master.groovy](#)
 - [Examples](#)
 - [Testing](#)
 - [How to find new FBC files - after a schema change](#)

tl;dr

After a schema change add upgrade rules to [master.groovy](#). Use the existing rules as examples.

FBC files and their purpose

Description

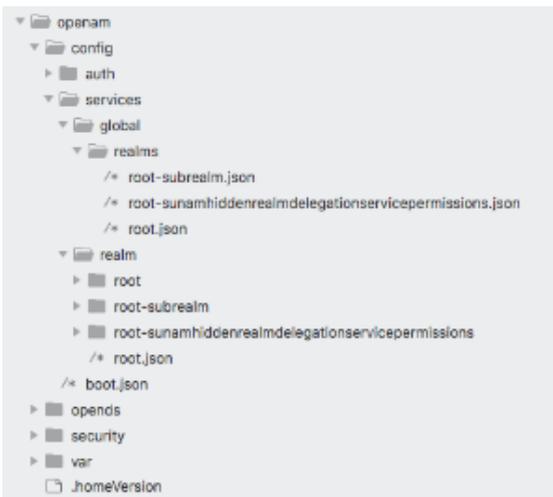
AM has the option of storing its configuration in

- LDAP, or
- File base

Other contents should be held in external data stores e.g. CTS, User store, applications and polices stores.

Structure

By default in `~/openam/config/services`

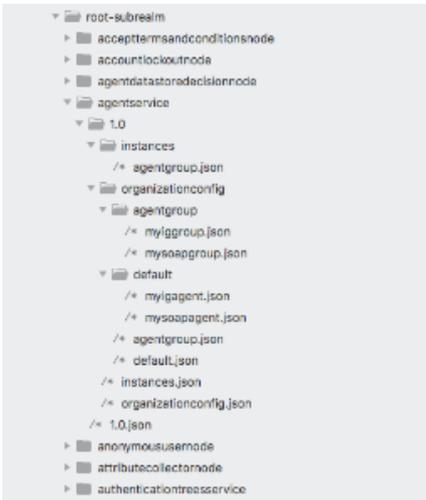


Above is the top level structure of the FBC. It has similarities with an Amster export but be aware that it is very different.

i Amster files != FBC files

N.B. Concentrating on the configuration files:

- `services/`
 - `boot.json`
 - `global/`
 - `realm/`
 - `root.json`
 - `root/`
 - `<custom realms>/`
 - `root-sunamhiddenrealmdelegationservicepermissions/`



Agent services for example

N.B.

- agentservice
 - 1.0.json
 - 1.0/
 - instances.json
 - instances/
 - agentgroup.json
 - organizationconfig.json
 - organizationconfig/
 - agentgroup.json
 - agentgroup/
 - <customgroup>.json
 - default.json
 - default/
 - <instance>.json

Rules to upgrade FBC

master.groovy

Set of idempotent rules that will upgrade file based configuration files to be compatible with the latest version of AM on a branch. [master.groovy](#) is contained in the AM.zip release.

 The rules within master.groovy must be idempotent.

Examples

Idempotency can be achieved using the configuration version. More information can be found here [Upgrade rule filtering based on version](#).

Idempotency using configuration version

```
def UPGRADE_TO_VERSION = "1.0.0.1"
def APPLICABLE_VERSIONS = ["1.1.0.0", "2.0.0.0"]
return
[
  setVersion(UPGRADE_TO_VERSION),
  forRealmService("authenticationTreesService",
    forVersionsBefore(APPLICABLE_VERSIONS,
      forRealmDefaults(
        addAttribute("new").with("attribute")
      )
    )
  )
]
```

From [authenticationTreesService-realm-defaults-add-attribute.groovy](#)

Guards can be used to achieve idempotency.

Idempotency using guards

```
return
[
  forRealmService("OAuth2Provider",
    forRealmDefaults(
      within("advancedOIDCConfig",
        where(key("authorisedIdmDelegationClients").isNotPresent(),
          addAttribute("authorisedIdmDelegationClients").with(Collections.emptySet()))),
      forSettings(
        within("advancedOIDCConfig",
          where(key("authorisedIdmDelegationClients").isNotPresent(),
            addAttribute("authorisedIdmDelegationClients").with(Collections.emptySet())))),
    ]
```

From [oauth2Provider-realm-instances-add-attribute-idempotent.groovy](#)

Testing

[Unit tests](#) will run against [master.groovy](#) for these [test cases](#). When adding rules it is expected that test case files are added to test those rules.

Further advice can be found in the associated [README](#).

How to find new FBC files - after a schema change

The following process could be followed to find out the contents of new or changed fbc as a result of a schema change, e.g. How to find the service name as it is not the Amster name.

- Make the schema change
- Install AM in FBC mode using this environment variable
 - `-Dcom.sun.identity.sm.sms_object_filebased_enabled=true`
- Examine the FBC in the AM configuration directory e.g. `~/openam/config/services``



The FBC service name != Amster service name