Running MBE

How to Install

Installation Files

Installation files (or called as artifact in Jenkins) come into 2 different forms:

1. `{applicationName}{version}{branch}{buildnumber}-conf.zip`. eg `petrolink-mbe-engine-1.0.0-rel1.0.0.33-conf.zip`. This file is essentially complete install of the system, pending configuration.
2. `{applicationName}{version}{branch}{buildnumber}-jar-with-dependencies.Jar`. eg `petrolink-mbe-engine-1.0.0-rel1.0.0.33-jar-with-dependencies.jar`. This file is core binary, which should be replaced during upgrade, if configuration file is similar. See optional step on upgrading.

Step 1. File copying

1. Copy the Jar or zip file and all configuration files in a folder. In Linux standard install location is in `/opt/mbe/`. And extract the zip file Structure of the files should be like

```
/{installPath}/bin
/{installPath}/doc
/{installPath}/Flows
/{installPath}/Triggers
/{installPath}/petrolink-mbe-engine-x.x-jar-with-dependencies.jar
/{installPath}/smartnow.cfg.xml
/{installPath}/connections.xml
/{installPath}/logback.xml
etc....
```

Step 2. Configuring Files

1. You need to copy all files and folder inside `tree_cep_engine\Engine\src\main\resources` except the log file. Alternatively you can extract petrolink-mbe-engine-*-conf.zip to setup your folder easily.
2. You need to modify `connections.xml` to configure Service API and RabbitMQ connection accounts.

Optional Step. Installing in Linux as Service

For production use on a Linux server, MBE should be configured to run as a Linux service (Systemd), that is, as a daemon process. This has the following advantages:

- MBE can be automatically restarted when the operating system restarts.
- MBE is less likely to be accidentally shut down, as can happen if the terminal was manually started in is closed.
- Logs from the MBE JVM can be properly managed by the service.

Standard install location is in `/opt/mbe/`

1. Copy example mbe-linux-systemd.service in `bin` of package folder or in `tree_cep_engine\Engine\src\main\bin` to `/usr/lib/systemd/system/` or `/etc/systemd/system/` (the location depends on linux flavor and version)
2. Edit the file above to ensure correct java Service. Main thing you may need to change is parameter :

```
ExecStart=/usr/bin/java -Duser.dir=/opt/mbe -DlogPath=log -Dmq.data=amq -ea -Xmx2G -jar /opt/mbe/petrolink-mbe-engine-{ver
```

> Pay attention to `-Xmx2G` which is Java parameter on specifying maximum memory and `/opt/mbe/petrolink-mbe-engine-{version}-jar-with-dependencies.jar` which is location of the jar file. You may need to update to correct one if you are upgrading

3. The enable the service to start at boot time by running the following in a terminal

```
systemctl enable mbe-linux-systemd.service
```

4. Reload Daemon configurations (also needed if you modify the .service file)

```
systemctl daemon-reload
```

5. Restart the system, to check that service starts as expected. To check status of the server

   ```
   systemctl status mbe-linux-systemd
   ```

6. Manually Start and Stop the service

   ```
   systemctl start mbe-linux-systemd
   systemctl stop mbe-linux-systemd
   ```

Optional Step. Encrypting Config file

Script `mbe-windows-enccon.bat` and `mbe-linux-enccon.sh`are provided to encrypt current connections.xml in windows and linux respectively.

> For easier configuration you may simply modify from `connections.xml.enccon.example` provided since version 1.0.4
>
> Do NOT forget to delete connections.xml.orig when you don't need the original plain configuration file anymore

To do manually you need to ensure in connections.xml, the xml element you want to encrypt has attribute `encrypt="true"` and the root element to have `encypted="false"`. For example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xml>
<Connections encrypted="false">
  <Connection id="rmqLocal">
    <User encrypt="true">RabbitMQUsername</User>
    <Lock encrypt="true">RabbitMQPassword</Lock>
    <Host>10.20.27.221</Host>
  </Connection>
  <Connection id="petrovault">
    <URL>https://subdomain.petrolink.net/PetroVaultApplication/</URL>
    <User encrypt="true">ReplicatorUsername</User>
    <Lock encrypt="true">AccountPassword</Lock>
  </Connection>
  <Connection id="petrolinksmtp">
    <Host>mail3.petrolink.net</Host>
    <Port>25</Port>
  </Connection>
</Connections>
```

Will produce

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xml>
<Connections encrypted="true">
  <Connection id="rmqLocal">
    <User encrypted="true">nwLfre4Jfa2P1lNyBFjcJO2dS8jvdOXy27xXQXhQJWQ=</User>
    <Lock encrypted="true">wIEQDYi854I0ctjXT6BjcE+XdpAN1n86XF8HeOxt1Fg=</Lock>
    <Host>10.20.27.221</Host>
  </Connection>
  <Connection id="petrovault">
    <URL>https://subdomain.petrolink.net/PetroVaultApplication/</URL>
    <User encrypted="true">btNMM86/Bwg79JWp94nTX7hisUXxb/BAUr57tucfDgM=</User>
    <Lock encrypted="true">AIc4X9XYwhAfftuUKS67/FPN/YsAvZlo</Lock>
  </Connection>
  <Connection id="petrolinksmtp">
    <Host>mail3.petrolink.net</Host>
    <Port>25</Port>
  </Connection>
</Connections>
```

File and Folder Components

This section describe primary components of the folder where the application is Installed

Configuration files

These files is used by application to know where to point input and output for the application. 1. `connections.xml` where you will need to configure RabbitMQ connection setting and service API account. 2. `smartnow.cfg.xml` where you will need to configure custom service API and RabbitMQ connection path. generally not needed to be modified unless some service path is changed. Note that, when needed, RabbitMQ connection strings in `smartnow.cfg.xml` looks like : `amqp://uname:@hostname` 3. `logback.xml` where you can configure logging configuration

Local state files

These files created or needed by engine during runtime to represent changing state. 1. Folder `h2` where the in memory database store Database file 2. Folder `flows` where active rules is saved 3. Folder `wells` where well configuration is saved 4. Folder `log` where logging is saved by default (automatically made on fresh start) 5. Folder `templates` where active templates for notification is saved. For Mail configuration iin some template it may need to be modified manually here

Binaries and Template

1. File `root-app-context.xml` where Spring configuration is saved (most of you don't need to modify this as this is developer's configuration).

2. `{applicationName}{version}{branch}{buildnumber}-jar-with-dependencies.Jar`. eg `petrolink-mbe-engine-1.0.0-rel1.0.0.33-jar-with-dependencies.jar`. This file is binary files containing the install file

3. Folder `doc` where documents is stored

Services and batch files

Folder `bin` is used to store example on how to configure OS related function. For example:

- mbe-windows.bat : Running MBE as Java Application in Windows
- mbe-windows-stdio.bat : Running MBE as Java Applications but output is redirected to stdio.log in Windows
- mbe-linux.sh : Running MBE as Java Application in Linux
- mbe-linux-systemd.service : example of configuration for systemd (System Daemon) for Linux

  ```
  Beware that in upgrading, the above files may need to be modified to point correct JAR file
  ```

How to Build/Obtain Installation files (Zip and Jar)

Preparing the JAR File through Jenkins

This process will involve building through Jenkins system which is configured through **jenkinsfile** in root folder. You will need to go to `MBE-Engine` project in Jenkins and select branch you want. How the branch is setup please refer to README.txt in `ReposTimbergrove` but essentially you generally want

```
    rel/{version} - Prefix for release branches based on the develop branch. This is where last minute preparation will go for
```

You then need to click `Build Now` to build the artifacts (which is basically our install files), or alternatively simply use the Last Successful or different build version Artifacts. You should see about 4 artifacts each

1. `{applicationName}{version}{branch}{buildnumber}-conf.zip`. eg `petrolink-mbe-engine-1.0.0-rel1.0.0.33-conf.zip`. This file is essentially complete install of the system, pending configuration.
2. `{applicationName}{version}{branch}{buildnumber}-jar-with-dependencies.Jar`. eg `petrolink-mbe-engine-1.0.0-rel1.0.0.33-jar-with-dependencies.jar`. This file is core binary, which should be replaced during upgrade.
3. `{applicationName}{version}{branch}{buildnumber}-jar`. eg `petrolink-mbe-engine-1.0.0-rel1.0.0.33.jar`. This file is core binary without dependencies, not used for now.
4. `petrolink-mbe-structures-{version}`. eg `petrolink-mbe-structures-1.0.jar`. This file is the model structure binary.

Download the files you need. You only need first one (**-conf.zip**) for fresh install, and you only second one (**-jar-with-dependencies.Jar**) for upgrade (as long you modify the service file mentioned in Service file

Manual JAR File building process (For Developers)

To prepare installation file, there are two things to be prepared

1. Compiled Java Code in form of jar
2. Configuration files

This manual build will produce **SNAPSHOT** files as build version is not tagged by any build system.

Preparing JAR file

There are multiple ways to create this Java Archive File. First two options will build locally, while third option will create jar file through Jenkins.

Note that : in eclipse you may need to check `Resolve Workspace Artifacts` and `Skip Tests`

Option 1. Using Launch File

You need to select `MBE Build JAR.launch` in Eclipse and right click > run As > Build Jar Package. You can find the result in `ReposTimbergrove\tree_cep_engine\Engine\target\package`. Main JAR file is {applicationName}-SNAPSHOT-jar-with-dependencies.jar

Option 2. Using Maven

You need to build source code using maven with goal of

```
    clean compile package verify
```

You can find the result in `ReposTimbergrove\tree_cep_engine\Engine\target\package`. Main JAR file is {applicationName}-SNAPSHOT-jar-with-dependencies.jar