LF NETWORKING

LFN Developer & Testing Forum

BMC Simulator Demo

June 10, 2021 – Maciej Miś

Agenda

- Introduction
- Prerequisites
- ODIM integration demo
- Running BMC simulator from IDE
- Code walkthrough
  - Configuration
  - Simulator Runner
  - Tree Templates
  - Behaviours

Introduction

BMC simulator is built to ease development / testing effort of ODIM

- Technology used - Kotlin
- API - DMTF Redfish implemented to OPAF profile

Prerequisites

- Openjdk-11
- IntelliJ IDEA
  - Plugin: Kotlin

Project has been developed and tested on Ubuntu 18.04 LTS

ODIM Integration - demo

Step 1 – Generate certs

Configuring TLS

1. To configure TLS you need to provide BMC simulator with certificates. BMC Simulator keeps certificates in KeyStores. There are 2 KeyStores:
   - trustStore - contains certificates used to verify incoming certificates, required for server's MTLS and TLS in http client
   - keyStore - contains a certificate used by BMC Simulator to identify itself.
2. Copy `rootCA.crt` and `rootCA.key` from `ODIM/build/cert_generator` or `/etc/odimracert/` to main BMC project directory (it should be the same file). If you are using BMC simulator in the test environment at localhost then BMC_DNS can be `localhost`. In a different situation you need to add `<bmc_ipv4_address> <bmc_dns>` in `/etc/hosts` in grf-plugin docker container. Use the following command to generate BMC TLS certificate and as argument use your BMC DNS:

```
$ ./generate_bmc_certs.sh <BMC_DNS>
```

ODIM Integration - demo

LFN NETWORKING
LFN Developer & Testing Forum

Step 2 – Prepare config file

- Set path for externalKeyStoreLocation and externalTrustStoreLocation
- Setup IP address for simulator (same IP address as indicated by BMC_DNS)

Example config file: `src/main/resources/odim-simulator-config.json`

ODIM Integration - demo

LFN NETWORKING
LFN Developer & Testing Forum

Step 3 - Build and run simulator

How to run standalone simulator

Prerequisites:

BMC simulator compilation and deployment requires Java 11. Simulator has been developed using OpenJDK with JRE11 and it was tested in such configuration.

1. Create executable jar

```
cd simulators
./gradlew executableJar
```

Run simulator with your config file:

```
java -jar simulator-runner-<version>.jar run BMC –c custom-config.json
```

ODIM Integration - demo

Step 4 - Register BMC simulator

If GRF has been turn on then you can send `POST` with registration simulator. If you are using containers remember to check whether BMC simulator is visible from a container with DNS which you used in `Configuring TLS` step. All new properties you can find in simulator configuration json file (exclude GRF_ENDPOINT_URL).

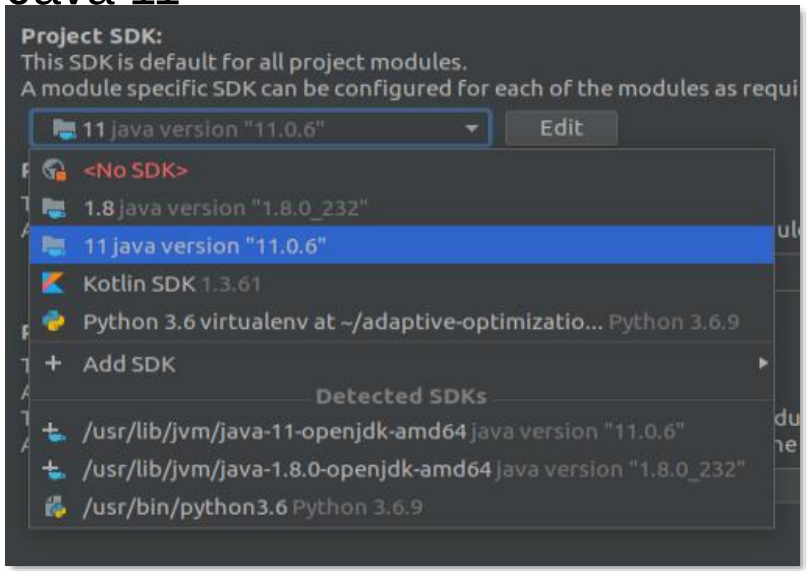
```
{
   "HostName": "<BMC_DNS>:<BMC_PORT>",
   "UserName": "<BMC_USERNAME>",
   "Password": "<BMC_PASSWORD>",
   "Links": {
       "ConnectionMethod": {
       "@odata.id": "<GRF_ENDPOINT_URL>"
       }
   }
}
```

Run and debug BMC Simulator

LFN NETWORKING
LFN Developer & Testing Forum

Using Intellij IDEA – basic setup

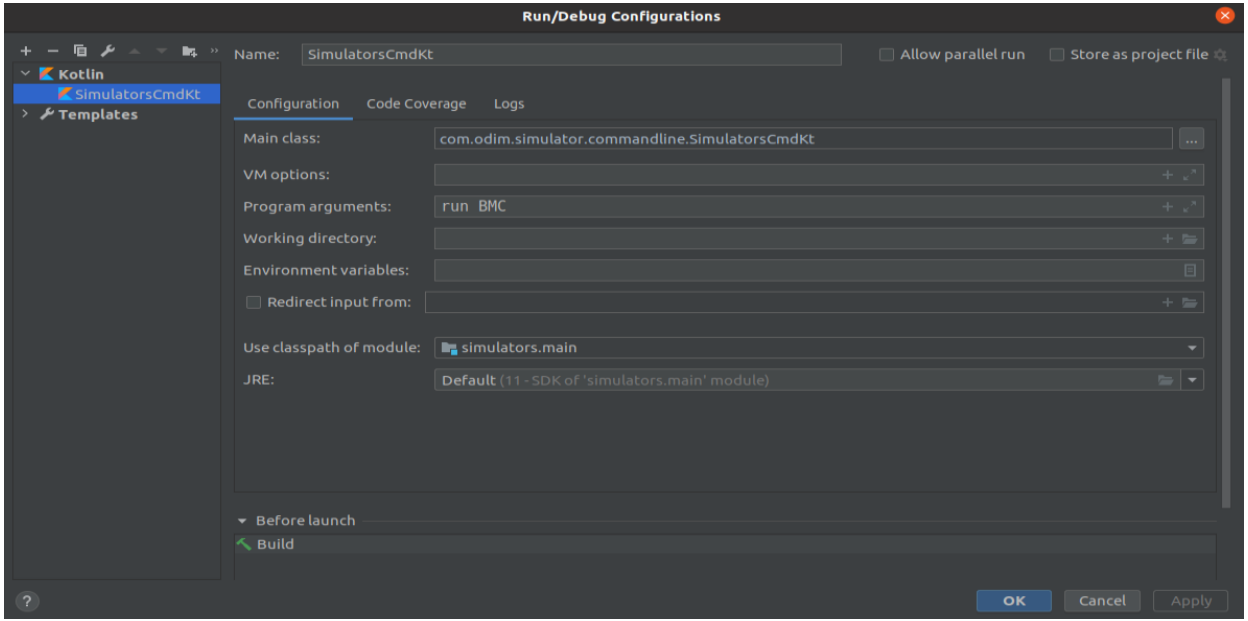
- ✓ Import project to IntelliJ
- ✓ Set project SDK in Project structure at Java 11
- ✓ Run in debug mode main function in SimulatorsCmd.kt



Run and debug BMC Simulator

LFN Developer & Testing Forum

Using Intellij IDEA – Run/Debug Configuration



Run and debug BMC Simulator

Logback

- Configuration file: `src/main/resources/logback.xml`
- Using logback configuration file you can set:
  - ✓ Appender – logging target, default STDOUT but can be set as FILE or both
  - ✓ Log format – default pattern:

```
%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n
```

  - ✓ Logging level– default setting: DEBUG

Full docs: http://logback.qos.ch/manual/configuration.html

Code walkthrough

JSON Config

Default config:

`simulator-config.json`

```
@Parameters(paramLabel = "SIMULATOR", description = ["Simulator to be run"])
private var name: String = ""

@Option(names = ["-i", "--ip"], description = ["Bind to ip address"])
private var ip: String? = null

@Option(names = ["-p", "--port"], description = ["Bind to port"])
private var port: Int? = null

@Option(names = ["-c", "--config"], description = ["External config file path"])
private var externalConfigFilePath: String? = "simulator-config.json"

override fun run() {
    logger.info("Running simulator with name: '{}' on port {}", name, port)
    appendValuesFromExternalConfig(externalConfigFilePath)
    val serveIp :String  = ip ?: getConfigProperty(SERVE_IP)
    val servePort :Int  = port ?: getConfigProperty(SERVE_PORT)
    serveSimulatorByName(name, serveIp, servePort)
}
```

Code walkthrough

Redfish endpoints mocked in simulator

```
211     private val storageController = create(STORAGE_CONTROLLER)
212     private val storage = create(STORAGE) { "Id" to "BMCStorage" }
213     private val bootOption = create(BOOT_OPTION)
214     private val ethernetInterface = create(ETHERNET_INTERFACE)
215     private val networkInterface = create(NETWORK_INTERFACE)
216     private val vlan = create(VLAN_NETWORK_INTERFACE)
217     private val pcieDevice = create(PCIE_DEVICE)
218     private val networkAdapter = create(NETWORK_ADAPTER)
219     private val networkPort = create(NETWORK_PORT)
220     private val networkDeviceFunction = create(NETWORK_DEVICE_FUNCTION)
221     private val logServiceForSystem = createLogServiceForSystem()
222     private val hostInterface = create(HOST_INTERFACE)
223     private val virtualMedia = create(VIRTUAL_MEDIA)
```

Code walkthrough

Tree and template examples

REST API is described by DSL
(Domain-Specific Language)

```
@Template(STORAGE)
open class StorageTemplate : ResourceTemplate() {
    init {
        version(V1_0_0, resourceObject( ...properties:
                "Oem" to embeddedObject(),
                "Id" to 0,
                "Description" to "Storage Description",
                "Name" to "Storage",
                "StorageControllers" to EmbeddedResourceArray(STORAGE_CONTROLLER),
                "Drives" to LinkableResourceArray(DRIVE),
                "Volumes" to ResourceCollection(VOLUMES_COLLECTION),
                "Status" to embeddedObject(STATUS),
                "Redundancy" to EmbeddedResourceArray(REDUNDANCY),
                "Links" to embeddedObject( ...properties:
                        "Oem" to embeddedObject(),
                        "Enclosures" to LinkableResourceArray(CHASSIS)
                ),
                "Actions" to Actions(
                        Action(SET_ENCRYPTION_KEY, parameterName: "EncryptionKey", mutableListOf())
                )
        ))
        version(V1_0_1, V1_0_0)
        version(V1_0_2, V1_0_1)
```

Code walkthrough

Behaviours

Behaviour is a code that runs on every call at selected endpoint

```
290            behaviors.prependBehavior(GET, securedBehavior())
291            behaviors.prependBehavior(POST, securedBehavior())
292            behaviors.prependBehavior(PATCH, securedBehavior())
293            behaviors.prependBehavior(DELETE, securedBehavior())
294
295            behaviors.replaceActionBehavior(RESET, ResetOnSystem(logServiceForSystem, bios))
296            behaviors.appendBehavior(COMPUTER_SYSTEM, GET, GetOnComputerSystem())
297            behaviors.appendBehavior(STORAGE, GET, GetOnStorage())
298            behaviors.prependBehavior(CHASSIS, GET, GetOnChassis())
```

```
33  class ResetActionBehavior : Behavior {
34      override fun run(tree: ResourceTree, item: Item, request: Request, response: Resp
35          val system :Resource  = (item as Action).parent as Resource
36          updateSystemBootProperties(tree, system)
37          return nonTerminal(noContent())
38      }
39
40      private fun updateSystemBootProperties(tree: ResourceTree, system: Resource) {
41          Merger.merge(tree, system, makeJson { this: DSL
42              "Boot" to {
43                  "BootSourceOverrideTarget" to null
44                  "BootSourceOverrideEnabled" to "Disabled"
45                  "UefiTargetBootSourceOverride" to null
46                  "BootSourceOverrideMode" to null
```


LF
NETWORKING
LFN Developer & Testing Forum