UeiBridge application new architecture.

Design, decisions, detailed specification.

Contents

[1 Use cases 2](#_Toc144386673)

[1.1 Use case 1: Accept new cube/rack 2](#_Toc144386674)

[1.2 Use case 2: Create new setup file 3](#_Toc144386675)

[1.3 Use case 4: Edit existing setup file 3](#_Toc144386676)

[1.4 Use case 4: UeiBridge startup 4](#_Toc144386677)

[1.5 Use case 5: UeiBridge closing. 5](#_Toc144386678)

# Use cases

## Use case 1: Accept new cube/rack

#### Description

When a new cube is accepted from vendor, new unique ip must be assigned to it and the cube details should be updated in global repository.

#### Actors

User.

PowerDNA explorer application.

CubeNet application.

#### Preconditions

The new cube is connected.

#### Postconditions

New ip assigned to new cube.

File CubeRepository.json (in ‘global folder’) is updated with details of the new cube/rack.

#### Steps

1. User launch CubeNet app
2. CubeNet tries to load repository file. On success: **IsRepExist=true**. On fail, only ‘create new rep’ and ‘load rep’ are available. Assuming user used one of them.
3. User clicks “***Fine free address for new cube***”, or, he can type the ip that he wants.
   * CubeNet checks in file CubeRepository.json what are the available ip’s.
4. CubeNet displays suggested ip.
5. User click ‘**Accept’**.
6. CubeNet show message whice asks user to use PowerDNA.

**File CubeRepository.json**

“CubeType”: {

SlotMap: “DIO403- SL508- SL508- SL508- SL508- DIO452”

NickName: “mcc”

TypeId: 1

“Desc”: “this cube should connect to mcc”

CubeList: [“192.168.100,3”, “192.168.100,15”, “192.168.100,17”]

}

“CubeType”: {

SlotMap: “AO308- DIO430- SL508- CAN503”

NickName: “ins”

TypeId: 3

“Desc”: “This cube should connect to ins”

CubeList: [“192.168.100,3”, “192.168.100,15”, “192.168.100,17”]

}

“CubeType”: {

SlotMap: “AO308- DIO430- SL508- CAN503”

NickName: “fcc”

TypeId: 4

“Desc”: “This cube should connect to fcc”

CubeList: [“192.168.100,21”]

}

Note the entry 2 and 3 has same slot map.

1. User clicks ‘ok’.
2. CubeNet locks the ip text box. **IsAddressLocked=true**.
3. User launce PowerDNA Explorer and sets the new ip for the new cube.
4. (use case might start at this point!)
5. User clicks “**get cube signature**”
6. CubeNet validates that the new cube is responding, reads from cube the slot map and show to user.
7. User clicks: ‘**Append cube to repository**’
8. If the slot map is already defined in the repository in one of the cube types, the user is asked if he want to add the cube to one of the existing cube types. If yes, the new cube ip is added to selected cube type. User click ‘save’. *Use case end*.
9. If the slot map is not already defined, of the user wants to define new cube type, he is asked to fill nick name and description for new cube type.
10. User clicks “**Update cube details**”
11. CubeNet Add CubeType entry to file CubeRepository.json. (The ‘type-id’ is generated automatically).
12. User clicks ‘save’.
13. CubeNet saves file. Use case end.

## Use case 2: Create new setup file

#### Description

Creates setup file for specific cube type.

This setup file contains operational definitions (baud rate etc..) as well as communication definitions (Receive/Transmit ip/port)

#### Preconditions

Access to updated CubeRepository,json file in global folder.

(Access to cube not needed).

#### Postconditions

New setup file created. (example “CubeSetup.**fcc**.config”).

#### Actors

User

***CubeDesign*** application.

#### Steps

1. User launch ***CubeDesign***.
2. User clicks button **‘create new cube setup’**.
3. ***CubeDesign*** display list of cube types (line per cube-type entry in CubeRepository.json)
4. User select one entry from cube type list and click ‘ok’.
5. ***CubeDesign*** generates default config and populate fields.
6. User edits fields.
7. User click ‘save’.
8. ***CubeDesign*** saves file with predefined name (based on cube-type nick name, for example “CubeSetup.**ins**.config”).
9. Use case end.

## Use case 4: Edit existing setup file

#### Description

After user generated default setup file, he wants to edit it.

#### Preconditions

#### Postconditions

Setup file updated.

#### Steps

1. User launch **CubeDesign**..
2. User clicks button ‘open existing setup file’.
3. ***CubeDesign*** opens a dialog and let user select file (\*.config)
4. User select file and clicks ‘ok’.
5. CubeDesign populate fields according to file.
6. User edit fields.
7. User click ‘save’.
8. Use case end.

## Use case 4: UeiBridge startup

#### Description

***UeiBridge*** intermediate between Simulator and Uei cubes/racks.

#### Actors

User.

***UeiBridge*** application.

#### Preconditions

#### Postconditions

UeiBridge connected to cubes, up and running.

#### Steps

1. User launch ***UeiBridge***.
2. ***UeiBridge*** search for all cubes in local network and maps the busy[[1]](#footnote-1) ones.
3. ***UeiBridge***: For each free cube/rack in network:
   1. Find cube-type.
   2. If this cube-type already exists in running-cubes-table, skip to next cube.

**Cube8.token.json file:**

{

CubeIp: 192.168.100.8

CubeType: mcc (id=5)

RunOnHost: DESKTOP-J0U9F0S

LastUpdate: August 31, 2023 14:08:34

}

* 1. Create token file[[2]](#footnote-2) to indicate that this cube is busy
  2. Add this cube to running-cubes-table in state “Acquired”.

1. UeiBridge: for each entry in state “Acquired” in running-cubes-table:
   1. Find suitable[[3]](#footnote-3) setup file.
   2. If setup file found, load it and ‘open’ the cube. Change the entry status to “waiting for activation”.
   3. If setup file not found, delete token file and change state to “no setup”.
2. UeiBridge: for each entry in state “Waiting for activation” in running-cubes-table
   1. Activate the selected cubes.
   2. Add log line in global log location: " UeiBridge on machine <XYZ> activated cubes 192.168.100.3”.
3. UeiBridge: wait for incoming messages by multicast, and forward them to output devices
4. UeiBridge: read message from input devices and sends them through multicast to consumer.
5. Use case end.

Running-cubes-table

|  |  |  |  |
| --- | --- | --- | --- |
| **Cube ip** | **cube type** | **cube state** |  |
| 192.168.100.3 | ins | 3- waiting for activation | Cube is ready to run and waits to be started. |
| 192.168.100.22 | gps | 1 – Acquired | Cube is marked as busy, but not yet up and running. |
| 192.168.100.5 | mcc | 2 - no setup | Setup file not found for this cube-type. |
| 192.168.100.49 | aaa | 4 – In dispose | User asked to close cube, but still in dispose state. |

## Use case 5: UeiBridge closing.

#### Description

UeiBridge closing steps.

#### Actors

User.

UeiBridge application.

#### Preconditions

#### Postconditions

UeiBridge up and running.

#### Steps

1. In UeiBridge console view, user types ‘enter’.
2. UeiBridge start disposing all activated devices.
3. UeiBridge Delete files "cube<N>.token.json" per each cube in global folder.
4. UeiBridge add log line (in global log location) per each disposed cube: " UeiBridge on machine <XYZ> closed cubes 192.168.100.3.
5. Use case end.

1. 'busy' means that a file "cube<N>.token.json" exists in global folder [↑](#footnote-ref-1)
2. For example: cube13.token.json [↑](#footnote-ref-2)
3. ‘suitable’ means that there is a match of slot/device between the cube and the setup file. [↑](#footnote-ref-3)