OpenBox Dashboard Description

­­­

Table of Contents

[Introduction 4](#_Toc489629970)

[Primary components 4](#_Toc489629971)

[Network Monitor Component 4](#_Toc489629972)

[OBA Controller Component 4](#_Toc489629973)

[Operational Modes 4](#_Toc489629974)

[Production mode 4](#_Toc489629975)

[Simulation mode 4](#_Toc489629976)

[Primary components 5](#_Toc489629977)

[Network Monitor Component 5](#_Toc489629978)

[Description 5](#_Toc489629979)

[Components 5](#_Toc489629980)

[OBA Controller Component 7](#_Toc489629981)

[Description 7](#_Toc489629982)

[Components 7](#_Toc489629983)

[Operational modes 8](#_Toc489629984)

[Production Mode 8](#_Toc489629985)

[Description 8](#_Toc489629986)

[Simulation Mode 8](#_Toc489629987)

[Description 8](#_Toc489629988)

[Components 8](#_Toc489629989)

[Implementation 10](#_Toc489629990)

[OpenBox Controller Implementation requirements 10](#_Toc489629991)

[Network Information REST service 10](#_Toc489629992)

[Network Actions REST service 10](#_Toc489629993)

[Profiling component 10](#_Toc489629994)

[Technology Stack 10](#_Toc489629995)

[OpenBox Dashboard Implementation requirements 10](#_Toc489629996)

[Client Side 10](#_Toc489629997)

[Server Side 10](#_Toc489629998)

[Technology Stack 10](#_Toc489629999)

# Introduction

The OpenBox dashboard is used to examine and monitor a deployed network and to extend the functionality of the network by deploying and removing OBAs.

## Primary components

The dashboard consists of two primary components:

### Network Monitor Component

* View a deployed network topology, OBAs, OBIs, processing graphs and processing blocks within the OBAs and the OBIs
* Monitor events and alerts sent through the OpenBox controller NorthBound API and SouthBound API

### OBA Controller Component

* Send read/write requests to the data plane
* Remove OBAs
* Deploy New OBAs

## Operational Modes

The open box dashboard works in one of two modes:

### Production mode

* Works on a deployed network

### Simulation mode

* Works on a simulative network
  + Network topology builder
  + OBI builder
  + OBA builder
  + Custom Block Builder
  + Simulative network deployment – initialize production mode with the created network

# Primary components

## Network Monitor Component

### Description

Through the network monitor component users can:

* Examine the network topology
* Examine the deployed OBAs
* Examine the loaded OBIs
* Profile network events
* Examine the performance gain obtained by the merge algorithm

### Components Initialization

* Initialize the OpenBox Dashboard with a deployed network
  + Provide the OpenBox Controller’s base url

#### Topology

* View Network topology tree
  + View applications by segment
  + View OBIs by segment
  + View applications by OBIs

#### OBAs

* Drill down to OBA preferences
  + Configuration parameters
  + App Priority
  + Location Specifier
  + Processing graph
    - Drill down to each processing block
      * Configuration
      * Registered Lifecycle Event handlers
      * Registered Read/Write handles
      * Registered Alerts handlers
      * **Application Aggregator -** Show “canMergeWith” blocks:
        + Within the OBA processing graph
        + Across all loaded OBAs
* View application alerts
* **Application Aggregator**
  + Show merge input & output
  + Analyze merged blocks across loaded OBAs
  + Show performance gain

#### OBIs

* Currently running OBIs
* Drill down to OBI preferences
  + Capabilities
  + Current Processing Graph
  + Drill Down to each processing block
    - Read/Write workflow
      * list registered read/write handles and registrars (apps)
      * list all available read/write handles
    - Event/Alert Workflow
      * list alerts

## OBA Controller Component

### Description

The OBA controller component allows the user to change the network’s behavior deploying new OBAs and removing deployed OBAs.  
The OBA controller also allows the user to send read/write requests to the OBAs deployed on the data plane and view the result.

### Components Send read/write requests

Send read/write requests to OBAs deployed in the data plane

#### Remove OBAs

Remove OBAs from the network

#### Deploy a new OBA and load it dynamically

New OBAs can be deployed in one of the following two deployment options:

* Deploy a jar
  + Upload An OpenBox application jar  
    \*Note – all non-moonlight dependencies must be packed in the jar
    - Source files
    - External jar files
    - External files (e.g firewall.properties)
* Deploy a predefined OBA

# Operational modes

## Production Mode

### Description

In production mode the users can examine, monitor and make actions in an existing network through the network monitor component and the OBA deployment component.

## Simulation Mode

### Description

In simulation mode the users can

* Create a network topology and OBIs
* Simulate the deployment of OBAs on the mock network
* Examine the network behavior and configuration
* Simulate traffic using dump files.
* Apply all capabilities of production mode views on top of the mock network

### Components

#### Topology Builder

* Define the network topology
  + Select a predefined topology or
  + Create a new topology

#### Custom Block Builder

* Create a custom block
  + Set the module name
  + Upload a binary
  + Define translations
  + \*Check if capabilities are defined for custom blocks

#### OBI Builder

* Define OBI capabilities through one of:
  + Select from predefined mock OBIs
    - Configure mock OBI
      * Load
      * Logging configuration
  + Create a new mock OBI through one of:
    - provide the API base url
    - create a custom OBI
      * Select capabilities
      * Create the OBI and redirect to “Select from predefined mock OBIs” screen
* Define OBI location
* Update the network topology

#### Simulator

* Initialize the controller with the defined topology
* Initialize the mock OBIs
* Initialize production mode with the current network
  + Redirect to ready production mode
  + Users can create and deploy OBAs to the network using the “OBA Builder Tool”

#### OBA Builder

* Select from a predefined application
  + Select a name
  + Configure
  + Register event and alert handlers per processing block
  + Define functionality upon lifecycle events
  + Select a location
  + Simulate merge algorithm and display the result
    - Show performance gain
  + Deploy the application
* Create a new application
  + Select a name
  + Define configuration parameters
  + Show all available processing blocks
  + Construct a processing graph
  + Register event and alert handlers per processing block
  + Define functionality upon lifecycle events
  + Select a location
  + Simulate merge algorithm and display the result
    - Analyze merged blocks for the given processing graph
    - Analyze merged blocks across all currently loaded OBAs
    - Show performance gain
  + Create the new application
    - Redirect to “Select from predefined applications” screen where the user can configure and run their OBA

# Implementation

OpenBox Controller Implementation requirements  
The OpenBox controller would be extended by the following

### Network Information REST service

Report on the deployed network components – topology, OBAs, OBIs etc.  
This data is available in the memory of the controller application.

### OBA Controller REST service

* Receive read/write requests and send them through the SouthBoundAPI
* Deploy/Remove OBA

### Profiling component

Collect the data sent through the northbound API and the southbound API and store it in a database.   
A worker will periodically execute all possible read requests to the deployed OBAs and save the responses in a database

### Technology Stack

* Jersey web services
* MySQL database (for now)

OpenBox Dashboard Implementation requirements

The OpenBox dashboard requires both client side and server side development.

### Client Side

The client side includes the views for the components detailed in this document as well as http services that communicates with the dashboard’s backend.

### Server Side

The OpenBox dashboard’s backend is responsible for providing all data required by the client side views through a REST API as well as exposing an API for performing actions on a network.

The backend will query OpenBox Controller’s “Network Information REST” service in order to obtain information on the network.

The backend will query OpenBox Controller’s “OBA Controller REST” service in order to make changes in the network.

The backend will query the database populated by the OpenBox controller in order to obtain the network’s event log

### Technology Stack

* Frontend framework - angular2
* Graphs – D3 framework - <http://jsfiddle.net/MetalMonkey/JnNwu/>
  + Display processing graphs component
* MySQL database
* Spring Boot Java application (Decoupled from the controller)

## Mock OBI Implementation requirements

The current mock OBI does not simulate traffic on the given processing graph.  
we can simply pre-populate the database with mock data to be used by the dashboard for demo purposes or keep simulating the data plane responses and maybe add a bit of randomness.  
If we want to make it actually use the processing graph, simulate traffic using dump files and responding on read requests with the actual block’s response we need to implement it.