HPE Unified OSS Console

Development Guide

Release 2.3.0
Part number: JP699AAE
Edition: 1.0

Hewlett Packard Enterprise

Notices

Legal notice

Printed in the US

Trademarks

Contents

List of tables

List of figures

Preface

About this guide

This user guide document describes the HPE Unified OSS Console product.

Audience

Here are some recommendations based on possible reader profiles:

- Administrator
- Operators
- Value Pack Designer
- Dashboards / Views Designer
- Integrator and delivery teams
- Unified OSS Console Add-ons Developers

Software Versions

The term UNIX is used as a generic reference to the operating system, unless otherwise specified.

Product Version	Supported Operating systems
HPE Unified OSS Console V2.3.0	Red Hat Enterprise Linux Server release 6.5 Red Hat Enterprise Linux Server release 7.2

Table 1 - Software Versions

Associated Documents

The following documents contain useful reference information:

- HPE Unified OSS Console V2.3 – Release Notes
- HPE Unified OSS Console V2.3 – Installation Guide
- HPE Unified OSS Console V2.3 – User Guide
- HPE Unified OSS Console V2.3 – Integration Guide

Support

Please visit our HP and HPE Software Support Online Web site at https://softwaresupport.hp.com/ for contact information, and details about HPE Software products, services, and support.

The Software support area of the Software Web site includes the following:

- Downloadable documentation.
- Troubleshooting information.
- Patches and updates.
- Problem reporting.
- Training information.
- Support program information.

Chapter 1
Unified Console Framework

1.1 Overview

The HPE Unified OSS Console (UOC) solution is a data visualization software platform. It is a generic web framework that facilitates the integration of various software systems (OSS, IoT, NFV…), and provides modern, responsive and dynamic web dashboards that are able to run on any device (tablet, phone, desktop…) to represent synthetic, highly summarized views.

It does not intend to replace existing legacy graphical user interfaces (GUIs) from underlying systems. It aims more to present aggregated data or dimensions, like high level statistics or metrics , coming from various sources, various data servers within the same web client (or page), with rich real-time and interactive graphics. The data is displayed in "real-time" in that it can be updated automatically in matters of seconds or minutes, what makes it useful for Operation and Analysis dashboards. A lot of features help the end user to analyze and navigate through information and quickly focus on the right meaningful information and help the decision makers.

The Unified OSS Console supports multiple security profiles and dynamically customizes/adapts the graphical user interface and information displayed, depending on the user's roles.

The interface is also totally value pack driven, and includes analysis tools allowing to browse these value packs, select some data and define some filters. It becomes easy to navigate and display information coming from multiple sources on the same dashboard.

The platform also provides report as a service. It is possible to call an URL to generate and export a PDF Report.

The HPE Unified OSS Console supports business critical service operations and processes. Some typical use cases include:

- A single page dashboard to represent the overall status of the network.
- A consolidated operations portal for an entire OSS solution.
- Executive dashboards.
- Personalized and dynamic web dashboards, built through definition files, and easily customizable with no need to develop additional content.
- Statistics Reports

Users/viewers can be diverse:

- Network operations teams with several profiles and right to access to information
- Service quality monitoring teams analyzing several value packs in multiple dashboards
- Executives
- Customer care teams or help desk operators
- Etc…

Integrators are also able to extend the Unified OSS Console: they can add some new components called "add-ons".

Some add-ons are at the server side to connect the Unified OSS Console to external data servers. Other add-ons are at client side to extend graphical components (widgets, layouts, themes, menus, module…) and enhance the customer experience providing consolidated views at end user level.

Here is a list of existing out of the box integration done with Unified OSS Console:

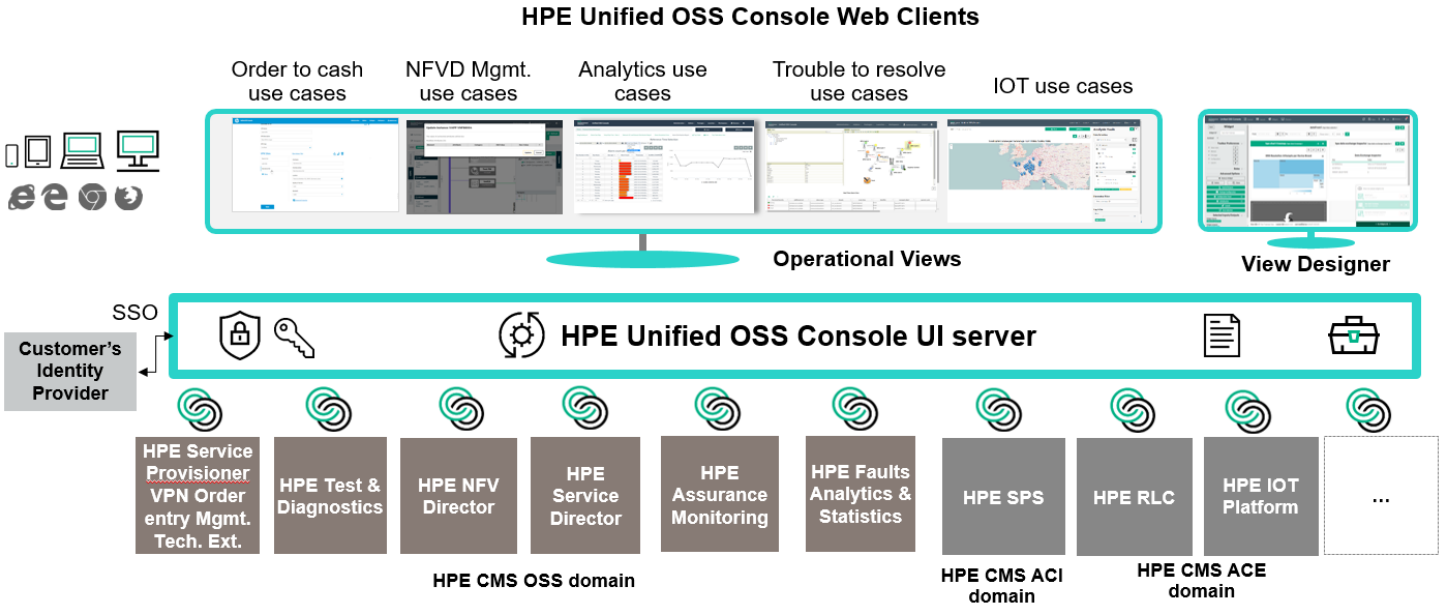


Figure 1 – Many out of the box Integration with Unified OSS Console

1.2 Architecture

The Unified OSS Console V2 supports a modern web architecture. It provides a flexible and open new user interface that support extensions (pluggable modules) to add features on the web browser (client) or integrate new domain servers (new one or existing legacy server).

The user interface is totally compliant with web standard (HTML5, CSS3 and JS), fully web responsive and can run on different devices and different display sizes. Themes are based on Bootstrap and allow a lot of customization capabilities.

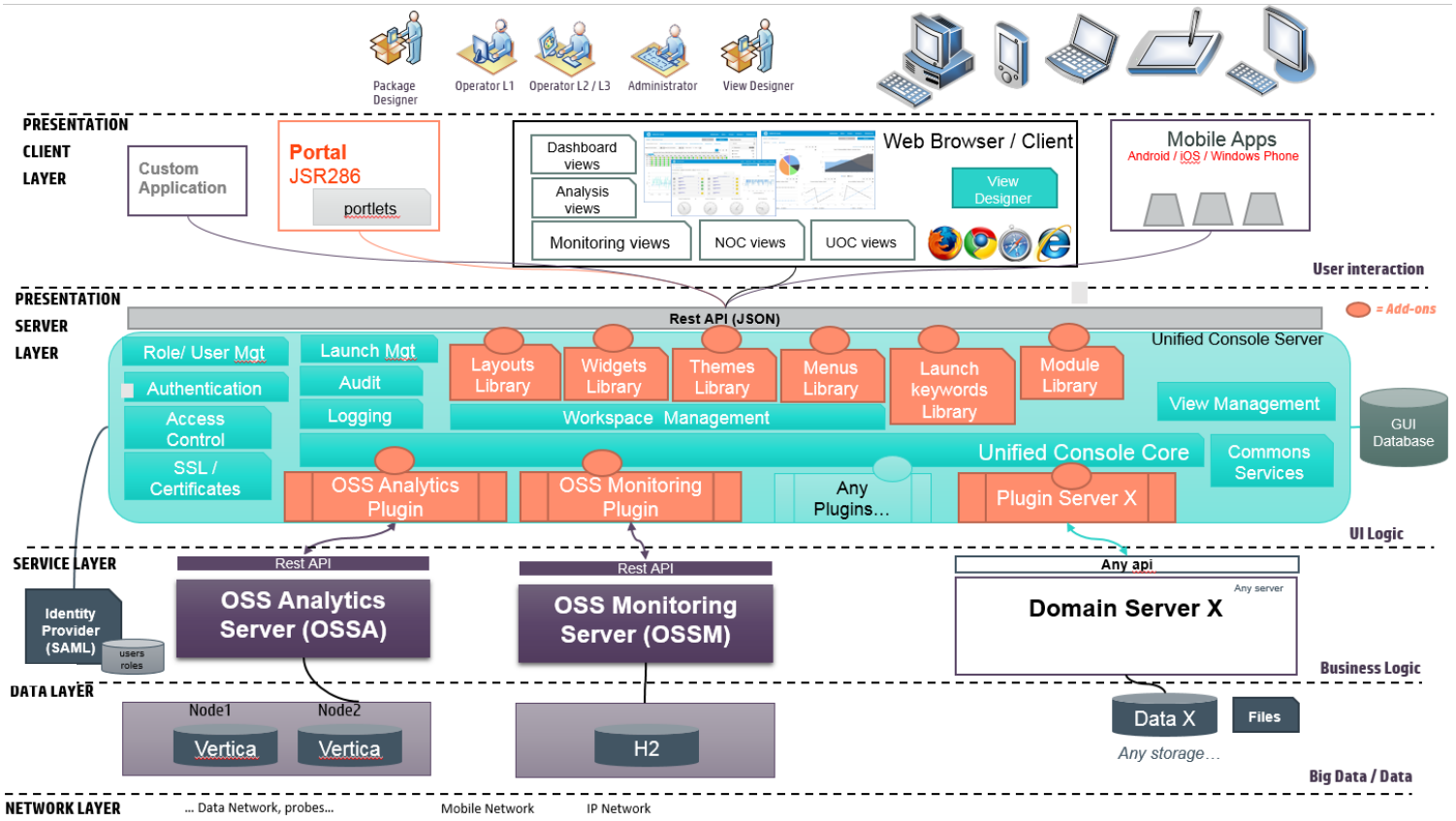


Figure 2 – HPE Unified Console Architecture

TIP: An add-on is an extension that can be dynamically installed on top of the Unified OSS Console and extend existing features (new widgets, new layouts, new menus, new data sources…). There are some add-ons at client side (graphical extension) or server side (plugin).

1.3 Modern Web Technology Stack

The Unified Console has been built on top of modern web technologies. Client-side and server-side components are both based on a JavaScript technology stack.

1.3.1 Full JavaScript Technology Stack

Integrators or developers who want to extend the Unified Console Framework will require to know these technologies.

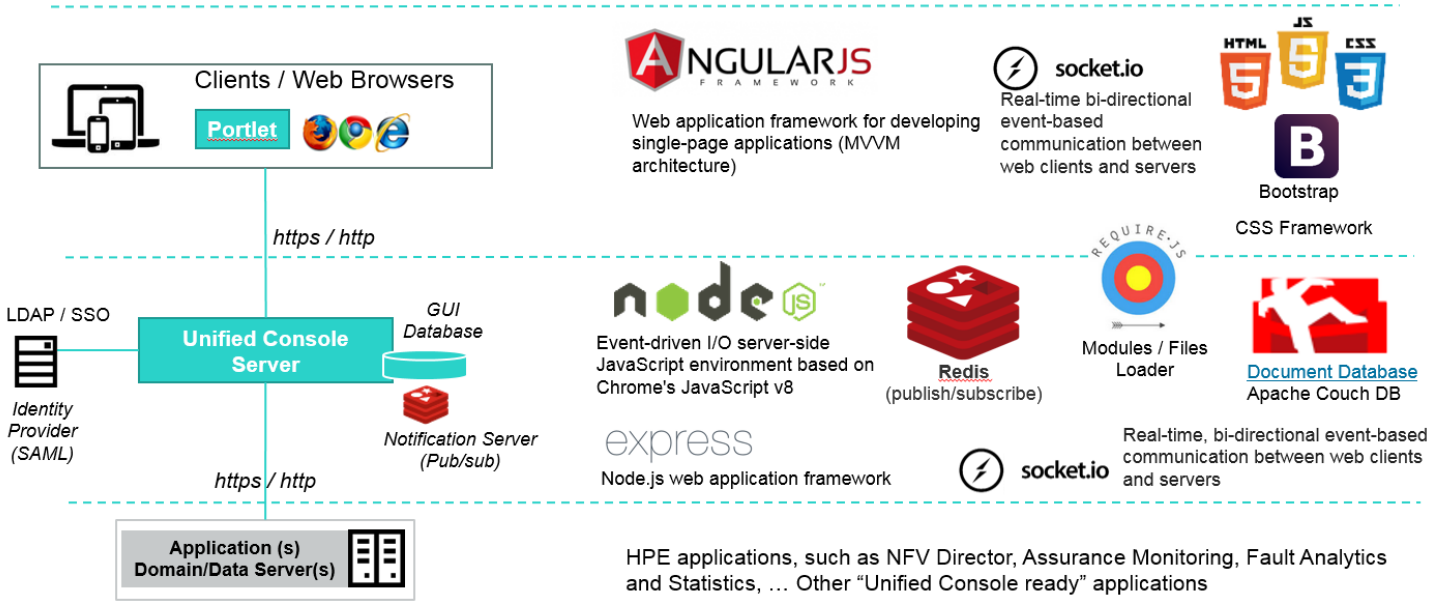


Figure 3 – HPE Unified Console Technologies

1.3.2 Server Side

- NodeJS: it is a JavaScript runtime platform built on top of Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient (https://nodejs.org)
- Express: It is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. (http://expressjs.com/)
- Passport: Passport is authentication middleware for Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based web application (http://passportjs.org/). Passport-SAML provide the SAML 2.0 authentication with Passport (https://github.com/bergie/passport-saml)
- Log4js: it is a conversion of the log4j framework to work with node (https://github.com/nomiddlename/log4js-node)
- RequireJS: it is a JavaScript file and module loader (http://requirejs.org/)
- Socket.io: Real-time bi-directional event based communication between clients and servers (http://socket.io/)
- Redis: It is an open source, in-memory data structure store, used as database, cache and message broker (http://redis.io/). Optionally used by Unified Console as a notification server.

1.3.3 Client Side

- Standard web technologies: HTML5, CSS3 and JavaScript
- AngularJS: AngularJS lets you write client-side web applications as if you had a smarter browser. It lets you extend the HTML syntax to express your application's components clearly and succinctly. It automatically synchronizes data from your UI (view) with your JavaScript objects (model) thanks to a 2-way data binding mechanism. To help you structure your application better and make it easy to test, AngularJS teaches the browser how to do dependency injection and inversion of control (https://angularjs.org/).
- Twitter Bootstrap: Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web (http://getbootstrap.com/)
- RequireJS: it is a JavaScript file and module loader (http://requirejs.org/)
- Socket.io: Real-time bi-directional event based communication between clients and servers (http://socket.io/)

1.3.4 Document Database

The GUI document database is a personal database dedicated to store graphical definition useful for UOC (workspaces, views, roles, permissions…).

- Apache CouchDB: it is a database that uses JSON for documents, JavaScript for MapReduce indexes, and regular HTTP for its API (http://couchdb.apache.org/). It also provide advanced features to ease replication and supports high availability.

IMPORTANT: This database never stores, computes or returns data value for dashboard or views. This database is dedicated to the Unified Console storage only.

1.3.5 Notifications Server

The notification server allows real-time bidirectional event-based communication between clients and servers in the case there are multiple UOC servers.

This notification server is based on Redis pub/sub:

NOTE: Redis pub/sub: Redis is an open source (BSD licensed), in-memory data structure store, used as database, cache and message broker. Redis includes a Publish/Subscribe messaging system that has been combined with Socket.io to allow real-time communication across multiple servers.
Only one Redis server is used by all UOC servers. (http://redis.io)

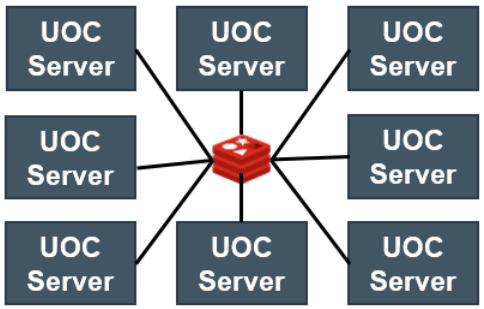


Figure 4 - UOC Servers and Redis (Notification Server)

IMPORTANT: Only one Redis server must be used by all UOC servers. The Notification server acts as a hub for all instance of UOC in chrge of distributing real-time notifications.

The notification server is an optional component in the solution. You need first to check how many UOC Servers will be running in your solution.

- ⇨ It could be an optional component in the solution if there is ONLY 1 UOC Server. Redis configuration should be disabled (active: false)
- ⇨ It is a mandatory component if there are multiple UOC server instances. In this case, Redis is used to allow communication between clients connected on different servers. Redis pub/sub is used as a common communication channel.

In the case multiple UOC server instances are installed, Redis must therefore be installed on the system.

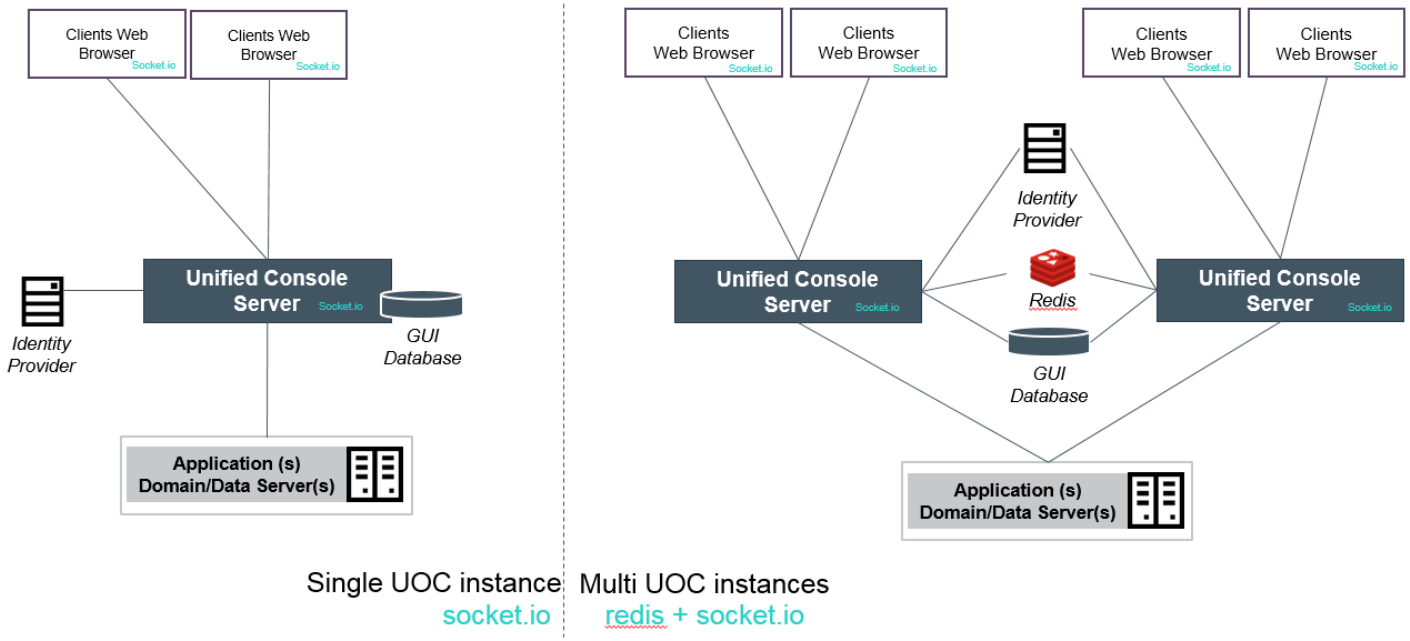


Figure 5 - UOC Servers and Redis – Deployment options (1 UOC Servers vs N UOC Servers)

1.4 Unified Console Add-ons

Add-ons are pluggable, reusable components used to extend the Unified Console and provide new capabilities. All these extensions are scanned dynamically at UOC server startup. The UOC platform is always up and running with the latest extensions available. There are 3 types of add-ons:

- Add-ons client: components running in the web browser like layouts, widgets, menus, etc....
- Add-ons server: Today, it is only a plugin component in charge of connection between UOC and data server(s) used by the domain.
- Add-ons data: All the JSON definition used by UOC engines to provide the feature to the end user operator (workspaces, views, launches,,etc.). Typically the View Designer editor is able to define graphically Views based on available add-ons clients and servers.

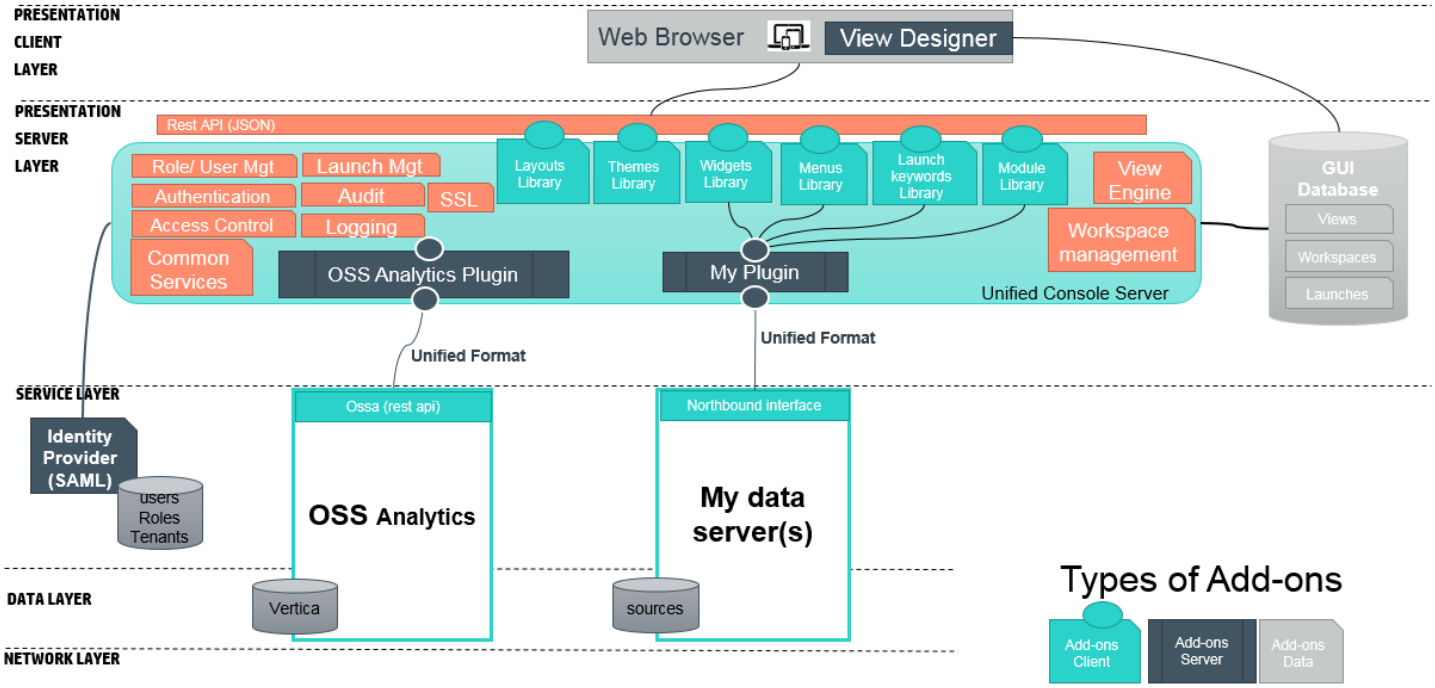


Figure 6 – Unified Console Add-ons

A platform administrator can browse these extensions with the add-ons menus (theme gallery, menu-bar gallery, widget gallery, layout gallery, plugins gallery, code module gallery) that will let the administrator explore all available add-ons on the platform and get additional information about them (name, description, version,...).

A developer can build his own add-ons and add new features on the UOC platform. These add-ons need to follow some development guidelines, implement mandatory APIs and provide a descriptor file in the right directories (JSON format).

⚠ **CAUTION:** All Unified Console Add-ons dedicated to be installed on a customer platform must be registered to the HPE Support Team. This id guarantees that no conflict can be raised in a running platform when multiple add-ons are installed.

An add-ons identifier is a unique string validated by the **support team (** https://softwaresupport.hp.com/**)** and listed in an official Add-on list.

1.5 How to build a solution?

Here the main steps to start building a solution based on Unified Console:

1. *Check your **data source(s)** i.e. domain server(s) and define your **package(s)** (data definitions exposed to GUI)*
2. *Select an existing or develop your **plugin** (leveraging the northbound interfaces of your data servers) or use the simulator to start. Plugin Simulator is a powerful tool that ease the development of demos, or Proof of concepts, etc. and does not require any connection to a data server.*
3. *Select existing client add-ons to **build your screens with View Designer** and group them in workspaces. You also can tune fine grain the security based on the roles of the connected users to generate the GUI based on his profile.*
4. ***Develop** missing components (if any) and **extend** add-ons library. You will need to use the SDK kit to get a proper add-on project structure and this SDK integrates a lot of development tools, especially add-ons generators to ease the building of a working skeleton and let you directly focus on the implementation of the content of your component and your own use case.*

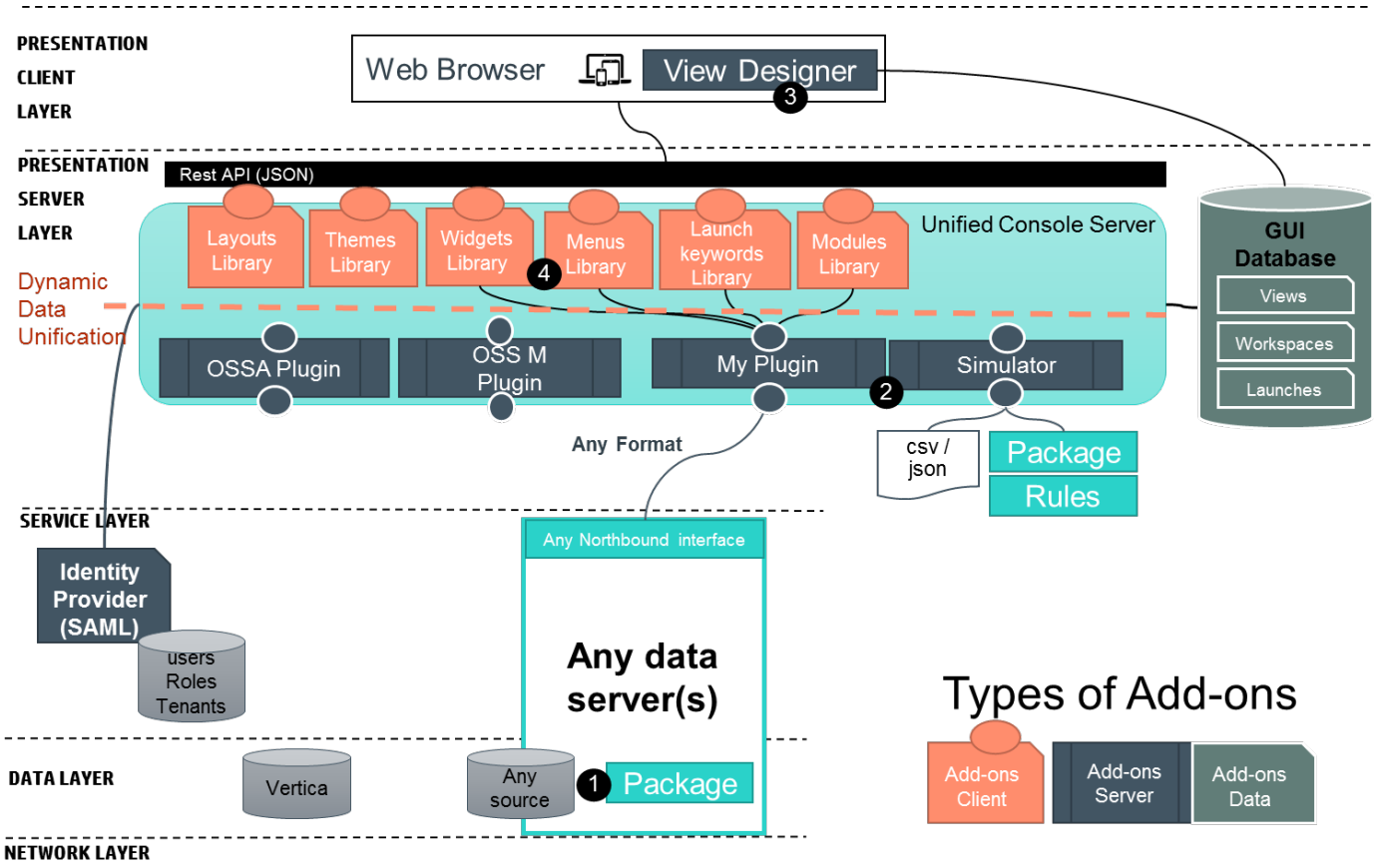


Figure 7: How to build a solution based on UOC foundation

TIP: To ramp up on the Unified Console, there are 2 trainings today available that will help you to quickly ramp up on the Unified Console concepts and developments.:

- Integration Training (3 days) that present an overview of UOC and describe all the mandatory concepts you need to understand to leverage all the features of UOC.
- Add-ons Development Training (5 days) that details all the add-ons and show you how to develop them to extend UOC with your own component and reuse them to build advanced solutions.

1.6 Unified Console API

Unified Console provides several API that is available to extends client and server add-ons. There are 4 types of APIs:

- Client Services : API dedicated for add-ons client (menu items, widgets, modules, etc.)
  - Common Events (see 18.1)
  - Data Exchange Services. (see 18.2)
  - Data Access Services (see 18.3)
  - Actions Services (see 18.4)
  - User Preferences Services (see 18.5)
  - Language Configuration Services (see 18.6)
  - Launches Services (see 18.7)
  - Map Configuration Services (see 18.8)
  - Theme Configuration Services (see 18.9)
  - Message Notifier Services (see 18.10)
  - Package Access Services (see 18.11)
  - Permissions Services (see 18.12)
  - Audio Services (see 18.13)

- Server Services: API dedicated for add-ons server (plugin).
  - Common Plugin API (see 16.1)
  - Access Control API (see 16.2)
  - Resource Security Logs API (see 16.5)
  - Session Security Logs API (see 16.6)
  - Platform Monitoring API (see 16.7)
  - Platform Statistics API (see Error! Reference source not found.)
  - Notifications API (see 16.3)
  - Server Logs API (see 16.4)

- Server REST API: Public interfaces provided by UOC. All features available on the client uses this interface.
  - Platform Monitoring (see 17.1)
  - Plugins (see 17.2)
  - Layouts (see 17.3)
  - Widgets (see 17.4)
  - Roles (see 17.5)
  - Users (see 17.6)
  - Permissions (see 17.7)
  - Workspace Categories (see 17.8)
  - Workspaces (see 17.9)
  - Languages (see 17.10)
  - Views (see 17.11)
  - Launch Categories (see 17.12)
  - Launches (see 17.13)
  - Launch Keywords (see 17.14)
  - Menu Bars (see 17.15)
  - Menu Items (see 17.16)
  - Modules (see 17.17)
  - Reports (see 17.18)
  - Export (see 17.19)
  - Themes (see 17.20)
  - User Preferences (see 17.21)
  - Publish / Subscribe (see 17.22)
  - Platform Statistics (see 17.23)

- Plugin REST API: Specific REST API available to query the plugin. This REST API is the official interfaces used by all add-ons client i.e. widgets to retrieve packages, query data, perform operations, etc.
  - Common (see 8.5.1)
  - Dimensions / Facts (see 8.5.2)
  - Objects / Operations (see 8.5.3)
  - Self-Monitoring (see 8.5.4)
  - Packages (see 8.5.5)

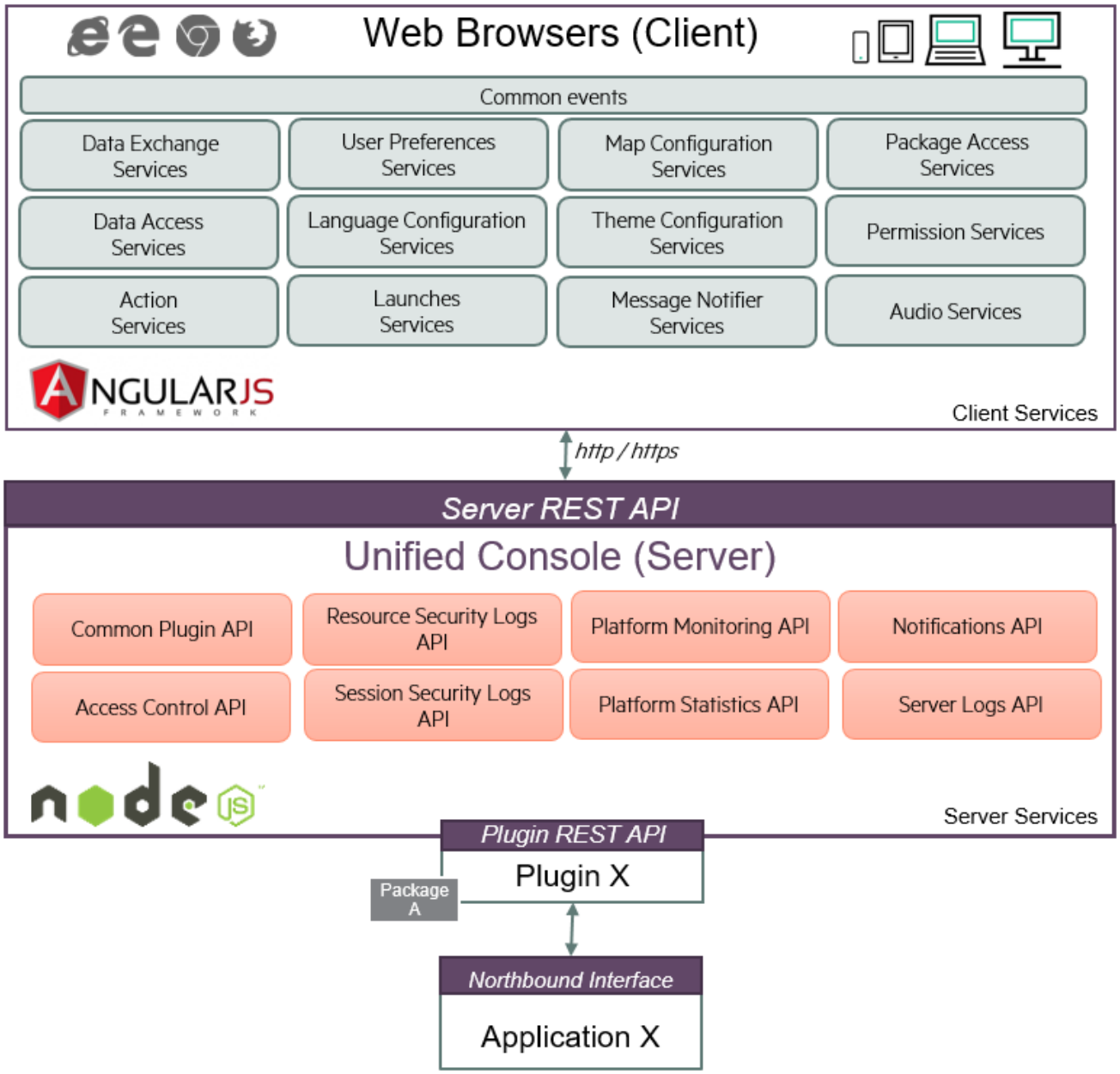


Figure 8: Unified Console API

Chapter 2
Angular Framework leveraged by UOC

Angular is a powerful JavaScript framework for building dynamic web applications. It became insanely popular nowadays. The good thing about Angular is that it has a set of ready-to-use modules to simplify building of single page applications. Unified Console fully leverage the modular architecture and the Single Page Application concept.

2.1 Single Page Application (SPA)

Single page application (SPA) is a web application that fits on a single page. All your code (JS, HTML, CSS) is retrieved with a single page load. And navigation between pages performed without refreshing the whole page.

When you are using SPA, you don't need to refresh the whole page, just load the part of the data which needs to be changed and render dynamically the view on the client side. It feels like a native application: mobile friendly, powerful, fast and responsive.

2.2 Model View – View Model

Model–view–view-model (MVVM) is a software architectural pattern. It facilitates a separation of development of the graphical user interface – be it via a markup language or GUI code – from development of the business logic or back-end logic (the data model). It is a variation of Model View Controller (MVC)

The view model is responsible for exposing (converting) the data objects from the model in such a way that objects are easily managed and presented. In this respect, the view model is more model than view, and handles most if not all of the view's display logic. The view model may implement a mediator pattern, organizing access to the back-end logic around the set of use cases supported by the view.

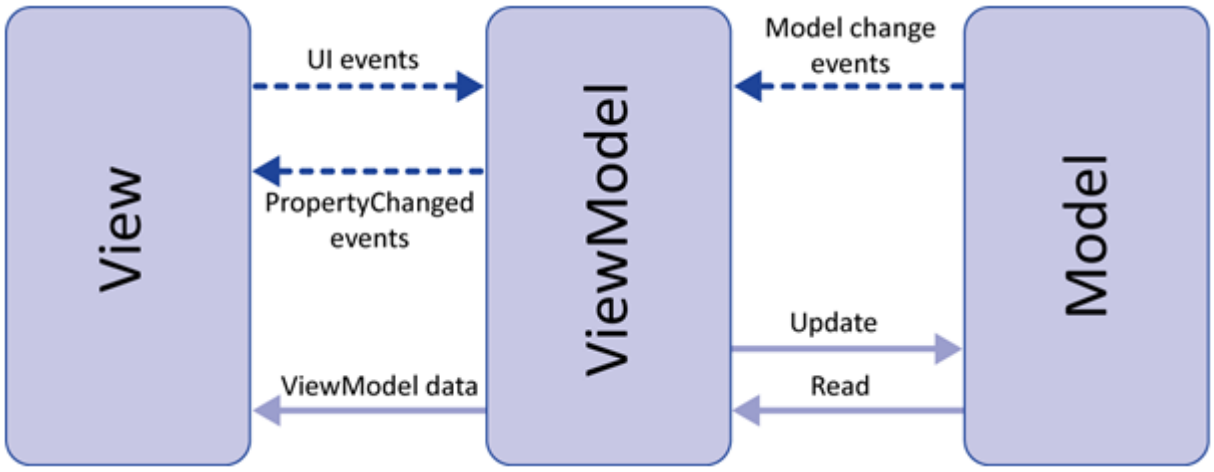


Figure 9: MVVM Architecture

There are 3 important points leveraged by UOC:

- MVVM is compatible with your existing MVC architecture.
- MVVM makes your apps more testable.
- MVVM works best with a binding mechanism.

2.3 Component-based MVC Framework

Angular is extremely modular and leverage the concept of components (or modules). A module is a container for the different parts of your application like controllers, services, filters, directives, etc. Everything in angular is a module (or components).

The key idea behind developing an Angular application is coming up with components and composing them together as required to build an application. A component is an independent cohesive block of code which has the required logic, view, and data as one unit. Each component will follow the Model View Controller (MVC) design pattern..

Components is an independent unit and they bring a lot of advantages:

- Highly reusable: you design and implement code as reusable modules.
- Totally independent: modules can be loaded in any order (or even in parallel). Coupled with a module loader (RequireJS) and a dependency injection mechanism, it bring very standalone components.
- High testability: Unit tests is very easy. Each module can be responsible of its own unit test. Angular test tools are integrated in the SDK to mockup and inject different scope for testing purpose.
- High maintainability: standalone module eases the refactoring preserving the interface with other modules (plug and play approach)
- Very scalable: scaling is as easy as adding more components to your existing application i.e. add more features.

UOC totally leverage this component based architecture, and allows extension of its UOC platform.

NOTE:. UOC foundation is built from 400+ angular modules. Your custom add-ons for the client side will be new module loaded at startup.

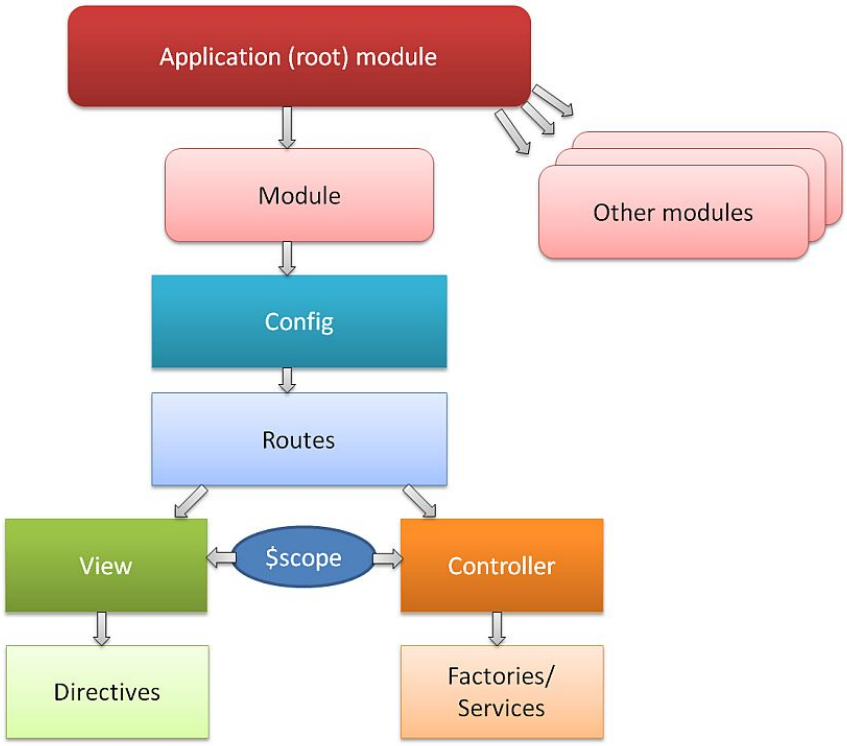


Figure 10: Angular Modules

IMPORTANT: Name the directory like the name of the module leveragin the addon identifier to make sure your component will be unique in UOC platform.

2.4 Data Binding

Data binding is the most useful and powerful feature of AngularJS. It is a process that bridges a connection between the view and business logic of the application. Below we see two ways of how data binding can take place – One being the traditional way, and the other with AngularJS.

There are One-Way or Two-Way data binding:

- One-Way (traditional) Data Binding : Data binding usually happens in one directions only, as shown in the figure below. It's a merged Template and Model into a View. Changes to the Model or related sections of the View are *not* automatically reflected in the View. Also, if user makes any changes to the View they are not reflected in the Model.

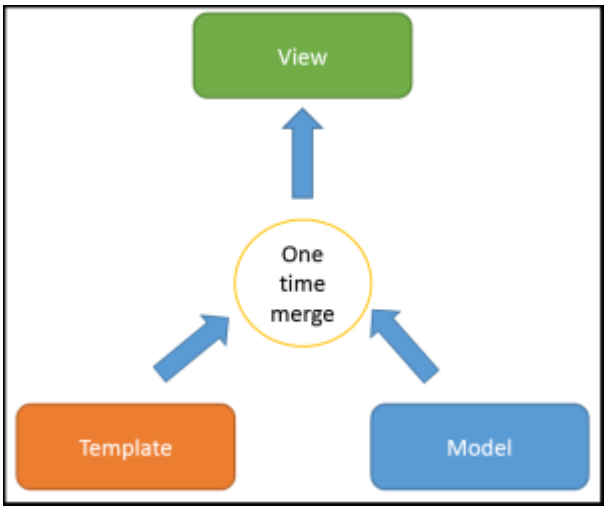


Figure 11: Traditional One-Way Data Binding

- Two-Way Data Binding (AngularJS): Any changes to the View are immediately reflected in the Model, and any changes in the Model are propagated to the View. The model is the single-source-of-truth for the application state, greatly simplifying the programming model for the developer.

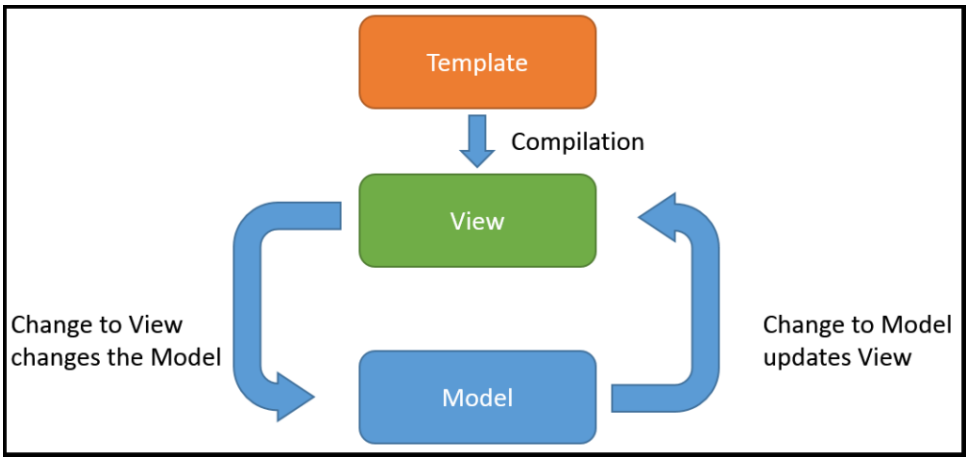


Figure 12: Angular Two-Way Data Binding

Please refer to the Angular documentation https://docs.angularjs.org/guide/databinding to get all the details of the powerful data binding.

2.5 Module Loader (RequireJS)

RequireJS is a JavaScript file and module loader. Please refer to the official website http://requirejs.org/ for details.

Require simplifies the module loading with the definition block at the beginning of each angular module. It will take care of loading if needed (not already done), the given module and cache it)

Example of require usage in an angular module.

RequireJS loads angular, loadash, commons event definition and the UOC service managing the configuration of the theme.

```
define([
    'angular',
    'lodash',
    'commons/events/commons-events',
    'components/theme-config/theme-config-services'
], function(angular, _) {
    'use strict';
    var xxxMyMenuItemControllers =
angular.module('xxxMyMenuItemControllers', ['commonsEvents',
'themeConfigServices']);
    xxxMyMenuItemControllers.controller('xxxMyMenuItemController',
['$scope', '$log', 'events', 'themeConfigService',
        function($scope, $log, events, themeConfigService) {
    // your code here
    ]);
    return xxxMyMenuItemControllers;
});
```

All code generators will use RequireJS to load correctly pre-defined services. I is mandatory to follow the same guidelines for your own custom components and avoid any issue with UOC.

IMPORTANT: Loading the correct file by requireJS and leverage the dependency injection to load a module are 2 different things. Both are mandatory to use a module, services, etc.

To make sure your module is standalone, all files needed by the dependency injection must be previously loaded. It will grant you that you do not hide a potential problem of loading, relying on a other module to load your own files.

2.6 Dependencies Injection (DI)

Dependency Injection (DI) is a software design pattern that creates and wires objects and functions.

The Angular injector subsystem is in charge of creating components, resolving their dependencies, and providing them to other components as requested.

Everything within Angular (directives, filters, controllers, services, ...) is created and wired using dependency injection. Within Angular, the DI container is called the injector.

You can use the dependency injection in the following ways:

- Components such as services, directives, filters, and animations are defined by an injectable factory method or constructor function. These components can be injected with "service" and "value" components as dependencies.
- Controllers are defined by a constructor function, which can be injected with any of the "service" and "value" components as dependencies, but they can also be provided with special dependencies. See below the recommended notation.
- The `run` method accepts a function, which can be injected with "service", "value" and "constant" components as dependencies. Note that you cannot inject "providers" into `run` blocks.
- The `config` method accepts a function, which can be injected with "provider" and "constant" components as dependencies. Note that you cannot inject "service" or "value" components into configuration.

It is recommended to use the inline array notation to inject services and make sure the minification processing will not impact your distribution code.

Example

```
someModule.controller('MyController', ['$scope', 'greeter',
      function($scope, greeter) {
  // ...
}]);
```

Here we pass an array whose elements consist of a list of strings (the names of the dependencies) followed by the function itself. When using this type of annotation, take care to keep the annotation array in sync with the parameters in the function declaration.

IMPORTANT: This inline array notation for dependency injection is mandatory to avoid any issue during the distribution processing, kit and packaging build.

2.7 Angular Translate (L10N)

All angular module can integrate very easily the localization mechanism provided by angular translate.

angular-translate provides a directive <translate> and a filter as components you can use to translate your apps content.

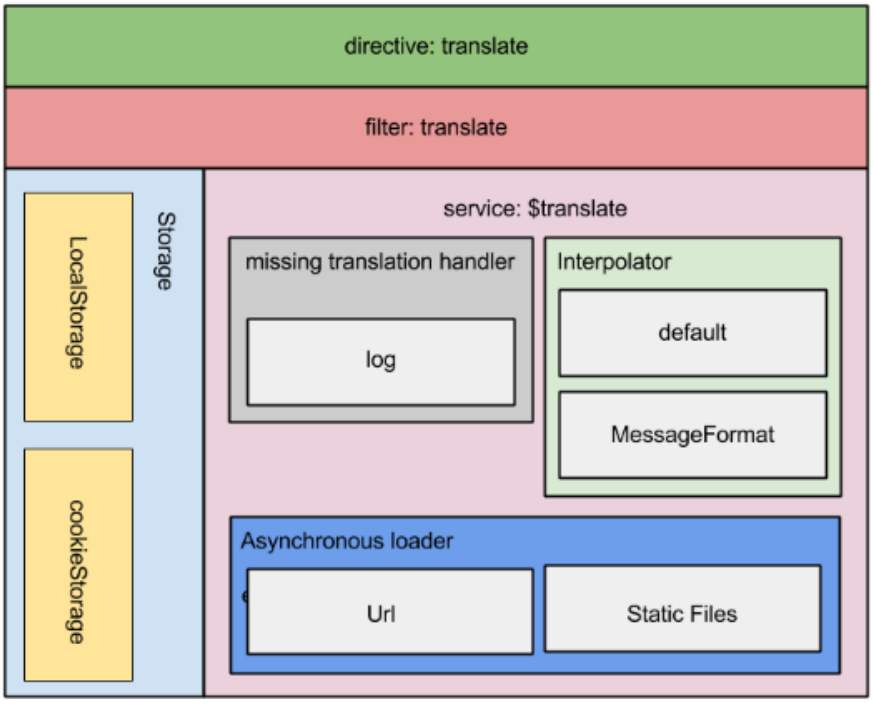


Figure 13: Angular Translate Conceptual view

NOTE:. See https://github.com/angular-translate/angular-translate and https://angular-translate.github.io for details of the localization management in angular module

```
'use strict';
define([
    'angular',
    'components/language-config/language-config',
], function(angular) {
    var xxxMyModule = angular.module('xxxMyModule',
['pascalprecht.translate']);


     xxxMyModule.run(['$translatePartialLoader',
          function($translatePartialLoader) {
             $translatePartialLoader.addPart('/addons/xxx/modules/xxx-
my-module');
          }
     ]);
    return xxxMyModule;
});
```

By default, it integrate support for angular translate through the directive **<translate>**

```
<div class="xxx-my-module">
      <p class="text-center">
            <translate>XXX_MY_MODULE.TODO</translate>
      </p>
</div>
```

2.8 Angular Directives

Directives are the most important components of any AngularJS application.

The directives file is in charge of implementing one or multiple angular directives for this module. Angular directives are markers on a DOM element (such as an attribute, element name, comment or CSS class) that tell AngularJS's HTML compiler ($compile) to attach a specified behavior. It means basically to extend the syntax of HTML with custom tags.

Please refer to the official documentation for more details: https://docs.angularjs.org/guide/directive

The directive defines the HTML tag, its HTML implementation and associated a controller. By default, all scope in the UOC directives are in isolated scope and exchange data through the directive parameters.

Directives definition has the following properties generated by UOC code generators:

- restrict – This provides a way to specify how a directive should be used in HTML. The legal restrict values are:
    - E for Element name. This is the default restrict value used by generator in UOC.
    - A for Attribute
    - C for Class
    - M for Comment

  Note: In this case we have set it to 'AE'. So, the directive can be used as a new HTML element or an attribute. Usually all directives used by UOC are restrict to element.
- templateUrl – This specifies the HTML markup from a separate HTML file that will be produced when the directive is compiled and linked by Angular.
- replace – This specifies if the generated template will replace the HTML element on which the directive is attached. In our case we have used the directive as <hello-world></hello-world>, and replace is set to true. So, after the directive is compiled, the produced output template replaces <hello-world></hello-world>. The final output is <h3>Hello World!!</h3>. If you set replace to false, the default, the output template will be inserted into the element on which the directive is invoked. Usually all directive in UOC will have replace equal to true.

The directive generated can be used with the following tag in HTML file. UOC usually invokes directive as element like:

```
<xxx-my-directive></xxx-my-directive>
```

IMPORTANT: You need to use the kebab case format in HTML i.e. All lowercase with - separating words.

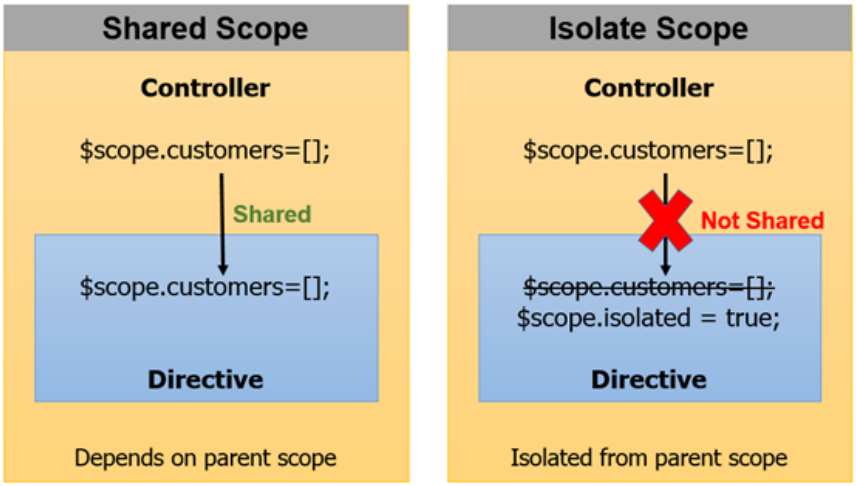
IMPORTANT: Naming convention is to postfix your directives file by **–directives.js** and defines one or multiple directives Your unit test file must be postfix by –directives-test.js

2.9 Angular Scopes

Scopes can be shared or isolated.

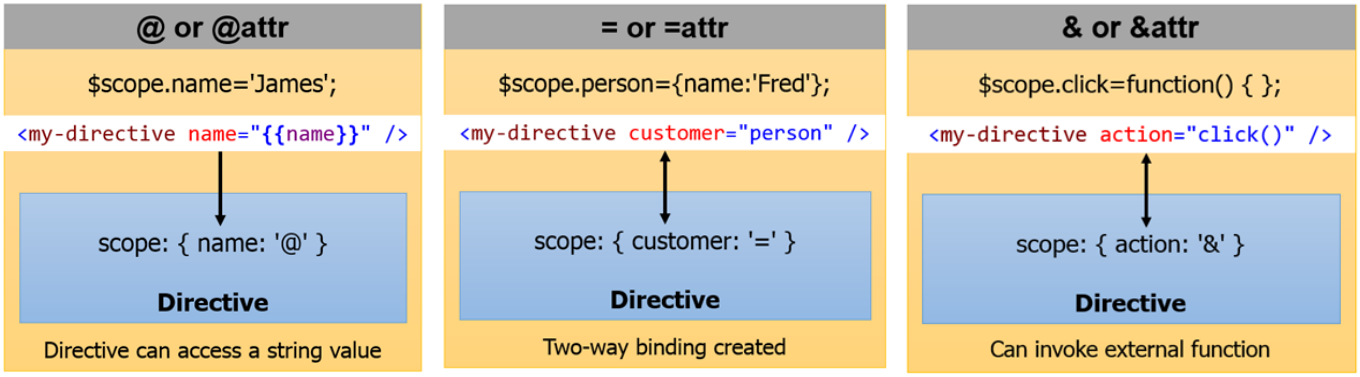
UOC generates and recommends to use isolated scope to protect your components and provide a loose coupling between components.

In Isolated scope, the directive does not share a scope with the controller; both directive and controller have their own scope.



However, data can be passed to the directive scope in three possible ways leveraging the local scope property.

1. Data can be passed as a string using the @ string literal
2. Data can be passed as an object using the = string literal
3. Data can be passed as a function the & string literal



Please refer to the angular documentation about scope to complete your knowledge on this important concept of angular: https://docs.angularjs.org/guide/scope and this very detailed WIKI: https://github.com/angular/angular.js/wiki/Understanding-Scopes

2.10 Angular Controllers

The controllers file is very important file, it is in charge of the interaction between the model (data) and the rendering part (view). Controller is basically a JavaScript constructor function that is used to augment the Angular Scope. When the directive is loaded, angular instantiate a new controller. A new child scope will be created and made available as an injectable parameter to the Controller's constructor function as $scope.

Please refer to the official angular documentation to get details on scope : https://docs.angularjs.org/guide/scope

The controller is always seen as the business logic for the view.

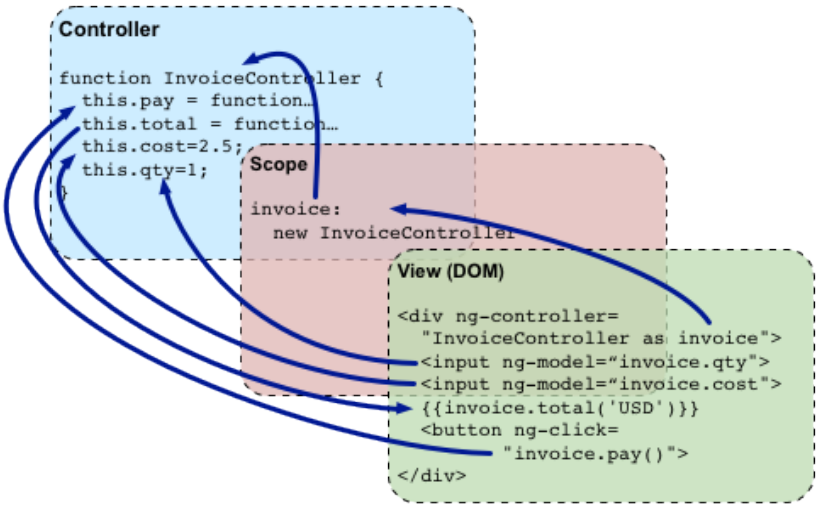


Figure 14: Angular Controller overview

IMPORTANT: Naming convention is to postfix your controllers file by **–controllers.js** and defines one or multiple controllers. Your unit test file must be postfix by –controllers-test.js

2.11 Angular Services

The services file is important in term of architecture and in charge of implementing one or multiple services for this module. Angular services are substitutable objects that are wired together using dependency injection (DI). You can use services to organize and share code across your app.

They always seen as View-independent business logic. Please refer to the official documentation for more details: https://docs.angularjs.org/guide/services

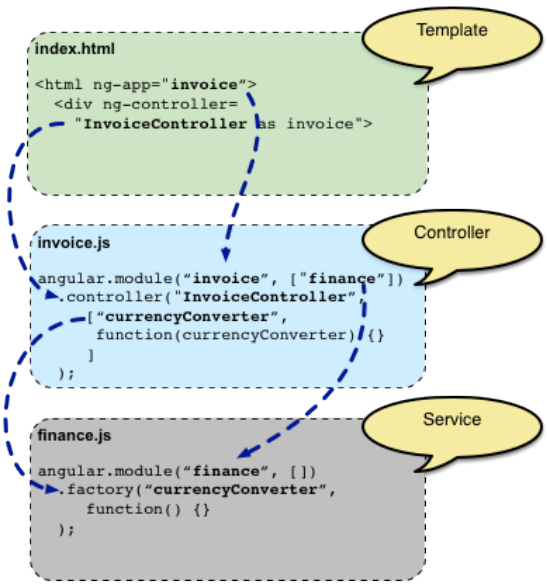


Figure 15: Angular Service overview

Angular services are:

- Lazily instantiated – Angular only instantiates a service when an application component depends on it.
- Singletons – Each component dependent on a service gets a reference to the single instance generated by the service factory.

IMPORTANT: Naming convention is to postfix your services file by **–services.js** and defines one or multiple services. Your unit test file must be postfix by -services-test.js

2.12 Angular Filters

Angular filters are in charge of formatting data. Angular provides some default filters, but you can write your own custom filters.

- **currency** Format a number to a currency format.
- **date** Format a date to a specified format.
- **filter** Select a subset of items from an array.
- **json** Format an object to a JSON string.
- **limitTo** Limits an array/string, into a specified number of elements/characters.
- **lowercase** Format a string to lower case.
- **number** Format a number to a string.
- **orderBy** Orders an array by an expression.
- **uppercase** Format a string to upper case.

These filters can be clubbed in expression or directives using pipe character.

Examples

This filter can be used in HTML or JavaScript.

```
{{ 'Hello world !' | uppercase }}
```

And also it can be used in JavaScript.

```
$filter('uppercase)( 'Hello world !')
```

Please refer to the official documentation for more details: https://docs.angularjs.org/api/ng/filter/filter and https://docs.angularjs.org/guide/filter

NOTE:. All data formatter in UOC rely on Angular filter modules (e.g. column formatter for widget hpe-ng-table,etc.)

2.13 Angular Logs

Angular provides a simple service for logging name **$log**. Default implementation safely writes the message into the browser's console (if present). The main purpose of this service is to simplify debugging and troubleshooting. UOC relies on this angular logs service for add-ons client. Please refer to the official documentation for more details: https://docs.angularjs.org/api/ng/service/$log

You also can log with a level of severity using a given methods:

- log
- error
- warn
- info
- debug

TIP: To open the web browser console of your favorite browser, you usually can press F12.

2.14 Angular Promises

In angular, there is a very powerful service $q that helps you run functions asynchronously, and use their return values (or exceptions) when they are done processing.

This is a Promises/A+-compliant implementation of promises/deferred objects inspired by Kris Kowal's Q.

NOTE: A promise represents the result of an asynchronous operation (e.g. a value that is not yet known). A promise is in one of three different states:

- pending - *The initial state of a promise.*
- fulfilled - *The state of a promise representing a successful operation.*
- rejected - *The state of a promise representing a failed operation.*

A deferred represents a chainable utility object with methods to register multiple callbacks into callback queues, invoke callback queues, and relay the success or failure state of any synchronous or asynchronous function.

Read more details about promise and deferred at http://documentup.com/kriskowal/q/

If you need to resolve multiple promises at once, this is easily achieved through **$q.all()** by passing in either an Array or Object of promises which will call .then() once both are resolved:

Please refer to the official documentation for more details: https://docs.angularjs.org/api/ng/service/$q

Chapter 3
Getting Started

3.1 Install runtime pre-requisites

You need to install the following pre-requisites:

- **Apache Couchdb**
- **NodeJS**
- **Redis (optional) – must be installed on Linux.**

3.1.1 Install NodeJS

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries (https://nodejs.org)

1. Install NodeJS for windows https://nodejs.org/en/
2. Click on 4.x LTS
3. Configure proxy for NPM
    a. Start a command line
    b. npm config set proxy http://my-proxy:8080
    c. npm config set https-proxy http://my-proxy:8080
4. Check NPM Configuration
    a. Start a command line
    b. npm config list

3.1.2 Install Apache CouchDB

Apache CouchDB is a distributed, fault-tolerant and schema-free document-oriented database (NoSQL),accessible via a RESTful HTTP/JSON API. Among other features, it provides robust, incremental replication with bi-directional conflict detection and resolution, and is queryable and indexable using a table-oriented view engine with JavaScript acting as the default view definition language.

A CouchDB database is a collection of documents; each document is a bunch of string "keys" and corresponding "values" (which can be numbers, strings, lists, dates ...). You can have indices, queries, views. (http://couchdb.apache.org/)

CouchDB offers us these features:

- Easy replication of a database across multiple server instances
- Fast indexing and retrieval
- REST-like interface for document insertion, updates, retrieval and deletion
- JSON-based document format

To install Apache CouchDB:

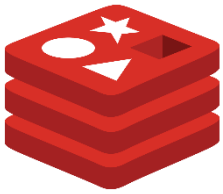
1. Install CouchDB for windows http://couchdb.apache.org/
2. Click on 'Download **1.6.1**' then 'Windows (x86)' and install the downloaded executable (defaults settings are fine)
3. After installation, open up Futon GUI (http://localhost:5984/_utils/)
    a. On the bottom right corner you'll find:
        – Welcome to Admin Party!
        – Everyone is admin. Fix this

b. Click the 'Fix This' link and create an admin user identified by
   - Username: admin
   - Password: admin

IMPORTANT: Do not use MS Internet Explorer. You may have some issue with Futon. It is recommended to use Firefox or Chrome.

NOTE: By default, Apache CouchDB is installed as a service on Microsoft Windows.

3.1.3 Install Redis



Redis pub/sub: Redis is an open source (BSD licensed), in-memory data structure store, used as database, cache and message broker. Redis includes a Publish/Subscribe messaging system that has been combined with Socket.io to allow real-time communication across multiple servers.

Only one Redis server is used by all UOC servers. (http://redis.io)

You need first to check how many UOC Servers will be running in your solution.

- ⇨ It could be an optional component in the solution if there is ONLY 1 UOC Server. Redis configuration should be disabled (active: false)
- ⇨ It is a mandatory component if there are multiple UOC server instances. In this case, Redis is used to allow communication between clients connected on different servers. Redis pub/sub is used as a common communication channel.

In the case multiple UOC server instances are installed, UOC relies on Redis 3.2.0 and above. A Redis server must therefore be installed on the system before installing UOC.

NOTE: As Redis does not support officially Windows, **You will need to install Redis on an external Linux machine, but most of the time, to develop an uoc addon, you will not really need such component.**

In case, you need a Redis server in your development, please follow the Installation Guide or get more details on their official web site: http://redis.io

3.2 Install Development Tools

To be able to extend the Unified Console, you need to install the following development tools:

- **Code quality check tools**
- **Testing tools**

Additional tools to ease the JavaScript development, testing and continuous integration

- **jsHint**
- **Unit testing (karma/mocha/chai)**
- **An editor/IDE that ease the code writing for JavaScript (SublimeText, MS Visual Code, WebStorm...)**

Several Web browsers to run the applications and troubleshoot your extensions (using the embedded developer tools). Debugging on the client will be done using the web developer tools.

TIP: We recommend Google Chrome for its good development tools (debugger, network activity...)

3.2.1 Install Git

Git is a free and open source distributed version control system (https://git-scm.com/). This is required for NPM and Bower to get released modules from the internet. It is strongly recommended to manage your addon source code project with Git.

1. Install Git for windows : https://git-scm.com/downloads
2. Click on 'Downloads for Windows' and install the downloaded executable
3. Configure proxy (https://git-scm.com/docs/git-config)
    a. Start a command line
    b. git config --global http.proxy http://my-proxy:8080
    c. git config --global https.proxy http://my-proxy:8080
    d. git config --global url."https://".insteadOf git://
4. Configure Git user information: User name and email (used for commit)
    a. git config --global user.name "myname"
    b. git config --global user.email "myemail@mycompany.com"
6. Check Git configuration
    a. git config --list

3.2.2 Node Package Manager (NPM)

npm is the package manager for JavaScript. Use npm to install, share, and distribute code; manage dependencies in your projects. The npm registry hosts over a quarter million packages of reusable code (https://www.npmjs.com)

NOTE: Node Package Manager is installed with Node JS. Please check this tool is available on your system after Node JS installation.

It is recommended to use npm 3.x version. Please check your npm version and upgrade your version if needed.

3.2.2.1 Version

Check that NPM is correctly installed on your system:

1. Start a command line
2. npm –version
3. check that the tools is installed and returned the installed version linked to the NodeJS installation.
4. npm help will display the commandline usage of the tool
    a. If you get a version below 3.x, please upgrade your npm. Execute npm install npm@latest –g to upgrade your node package manager.

```
npm –version
3.10.6
```

npm can return a lot more than just its own version - it can return the version of the current package, the Node.js version you are using and OpenSSL or V8 versions:

1. Start a command line
2. npm version

```
c:\UOC_Addon>npm version
{ 'hpe-unified-console-dev': '2.3.0',
  npm: '3.10.6',
  ares: '1.10.1-DEV',
  http_parser: '2.7.0',
  icu: '56.1',
  modules: '46',
  node: '4.5.0',
  openssl: '1.0.2h',
  uv: '1.9.1',
  v8: '4.5.103.37',
  zlib: '1.2.8' }
```

3.2.2.2 Help

As most cli toolkits, npm has a great built-in help functionality as well. Description and synopsis are always available. These are essentially man-pages.

```
c:\UOC_Addon>npm help

Usage: npm <command>

where <command> is one of:
    access, adduser, bin, bugs, c, cache, completion, config,
    ddp, dedupe, deprecate, dist-tag, docs, edit, explore, get,
    help, help-search, i, init, install, install-test, it, link,
    list, ln, logout, ls, outdated, owner, pack, ping, prefix,
    prune, publish, rb, rebuild, repo, restart, root, run,
    run-script, s, se, search, set, shrinkwrap, star, stars,
    start, stop, t, tag, team, test, tst, un, uninstall,
    unpublish, unstar, up, update, v, version, view, whoami

npm <cmd> -h     quick help on <cmd>
npm -l           display full usage info
npm help <term>  search for help on <term>
npm help npm     involved overview

Specify configs in the ini-formatted file:
    C:\Users\PicardJC\.npmrc
or on the command line via: npm <command> --key value
```

```
Config info can be viewed via: npm help config

npm@3.10.6 C:\Users\PicardJC\AppData\Roaming\npm\node_modules\npm
```

3.2.2.3 Install and saving dependencies

Once you found the package you want to include in your add-on project, you have to install and save it. The most common way of doing that is by using :

```
npm install <my npm package identifier>
```

usually you also want to automatically add it to your package.json file, so you can do:

```

npm install <my npm package identifier> --save
```

npm will save your dependencies with the ^ prefix by default. It means that during the next npm install the latest module without a major version bump will be installed.

This behavior is not recommended because the version can change over the time, so the add-on developer prefer to fix the version available at a time and make sure every install will use the same version. You totally control the required version for the package for your delivery.

To save the exact version (recommended), you need to enter::

```
npm config set save-exact true
```

3.2.3 Install Grunt

Grunt is a JavaScript Task Runner. It is used to automate any kind of repetitive tasks (minification, compilation, unit testing and linting…) by programming or using a command line.

To install grunt CLI (command line interface)

1. Start a command line
2. npm install -g grunt-cli
3. check the grunt commandline

```
grunt –-version
grunt-cli v1.2.0
grunt v1.0.1
```

3.2.4 Install Bower

Bower is a package manager for the web and it is optimized for front end (https://bower.io/)

NOTE: Bower requires Node JS, NPM and Git.

1. Install Bower with NPM
    a. Start a command line
    b. npm install -g bower
2. Configure the proxy for Bower
    a. Edit the file *.bowerrc*

```
{
   "directory": "client/bower_components",
   "proxy" : "http://web-proxy:8080",
   "https-proxy" : "http://web-proxy:8080",
   "strict-ssl": false
}
```

3.2.5 Install Node Sass

Sass (http://sass-lang.com/) is a CSS Preprocessor. It is a scripting language that extends CSS by allowing developers to write code in one language and then compile it into CSS. Sass is completely compatible with all versions of CSS.

Node-sass is a library that provides binding for Node.js to the popular stylesheet preprocessor Sass. It allows you to compile SCSS (.scss) files to CSS.

Check their official web site for details: https://www.npmjs.com/package/node-sass

To install node-sass

1. Start a command line
2. npm install -g node-sass
3. check the installation

```
node-sass –-version
node-sass       3.10.1   (Wrapper)       [JavaScript]
libsass         3.3.6   (Sass Compiler) [C/C++]
```

NOTE: This tool is only mandatory if you are planning to leverage Sass to generate your CSS for your components or to compile new themes. If you write directly the CSS for your components, there is no need to install this preprocessor.

3.2.6 Install Yeoman

Yeoman (or 'Yo') is a generic scaffolding system allowing the creation of any kind of app. It allows for rapidly getting started on new project. It runs as a command-line interface written for Node.js and which combines several functions into one place, such as generating a starter template, managing dependencies, running unit tests, providing a local development server, and optimizing production code for deployment (http://http://yeoman.io). Yeoman by itself doesn't make any decisions. Every decision is made by *generators*. Unified Console provides several generator to quickly build UOC add-ons (client and server side)

To install yeoman:

1. Install Yeoman with NPM
    a. Start a command line
    b. npm install -g yo

Yo also provide the following commands:

- yo --help Access the full help screen
- yo --generators List every installed generators
- yo --version Get the version

3.2.7 Install a Source Code Editor

There is no mandatory tool defined to create new components for Unified Console. It is up to the addons developer to select and use the editor he is used to work with.

Here after some recommendations working well with Unified Console.

3.2.7.1 SublimeText

Sublime Text is a sophisticated text editor for code, markup and prose. It provides a very good user interface, powerful editing features (multiple selection, split editing, snippets…) and great performance (https://www.sublimetext.com)

Sublime Text may be downloaded and evaluated for free, however a license must be purchased for continued use. Please refer to the website for subscription.

This editor allows installation on additional package to ease development and productivity, It is recommended to customize SublimeText with the following packages.

Sublime Test does not come with inbuilt package manager but there is a third party package manager called "Package Control". You can find more information about that on the following location: https://packagecontrol.io/

1. There is instruction provided for installation at the following link: https://packagecontrol.io/installation
2. Setup proxy settings for SublimeText.
    - Open "Preferences -> Package Settings -> Package Control -> Settings – Default" and edit
        - "http_proxy": "http://my-proxy:8080 ",
        - "https_proxy": "http://my-proxy:8080 ",

    It is recommended to install the following packages from the package manager. Type Ctrl+ Shift + P and select Package Control: Install Package, then select the following packages:
    - AngularJS
    - JSHint
    - BracketHighlighter
    - Emmet

- HTML-CSS-JS Prettify
- JsMinifier
- JSONLint
- Nodejs
- SublimeLinter

3.2.7.2 Jetbrains Web Storm

WebStorm is a lightweight and powerful IDE, perfectly equipped for complex client-side development and server-side development with Node.js (https://www.jetbrains.com/webstorm). It is not a free product, please refer to Jetbrains website for subscription.

3.2.7.3 Microsoft Visual Studio Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs.(https://code.visualstudio.com). It supports cross-platform development (OS X, Linux, and Windows)

Chapter 4
Development Environment (SDK)

4.1 Overview

The SDK provide ready to use environment to develop, run, check code, tests and build distribution kit for UOC add-ons. It leverage the best tools to ease live and rapid development and provide a continuous environment and provide extension like code generators.

It embed the following JavaScript tools to ease the development cycle.

- Grunt: The JavaScript Task Runner (http://gruntjs.com/)
- JSHint: A Static Code Analysis Tool for JavaScript (http://jshint.com/)
- CSSLint: A static Analysis Tool for CSS (http://csslint.net/)
- Live Reload: Middleware for adding the live reload script (https://github.com/intesso/connect-livereload)
- UglifyJS: JavaScript parser/compressor/beautifier toolkit (https://github.com/mishoo/UglifyJS)
- Karma: Test Runner for JavaScript (http://karma-runner.github.io/)
- Mocha: Test Framework for JavaScript (https://mochajs.org/)
- Chai: BDD / TDD assertion library for node and the browser (http://chaijs.com/)
- PhantomJS : a Headless webkit testing (http://phantomjs.org/)
- Yeoman:: Yeoman Project Scaffolder / Generator (http://yeoman.io)

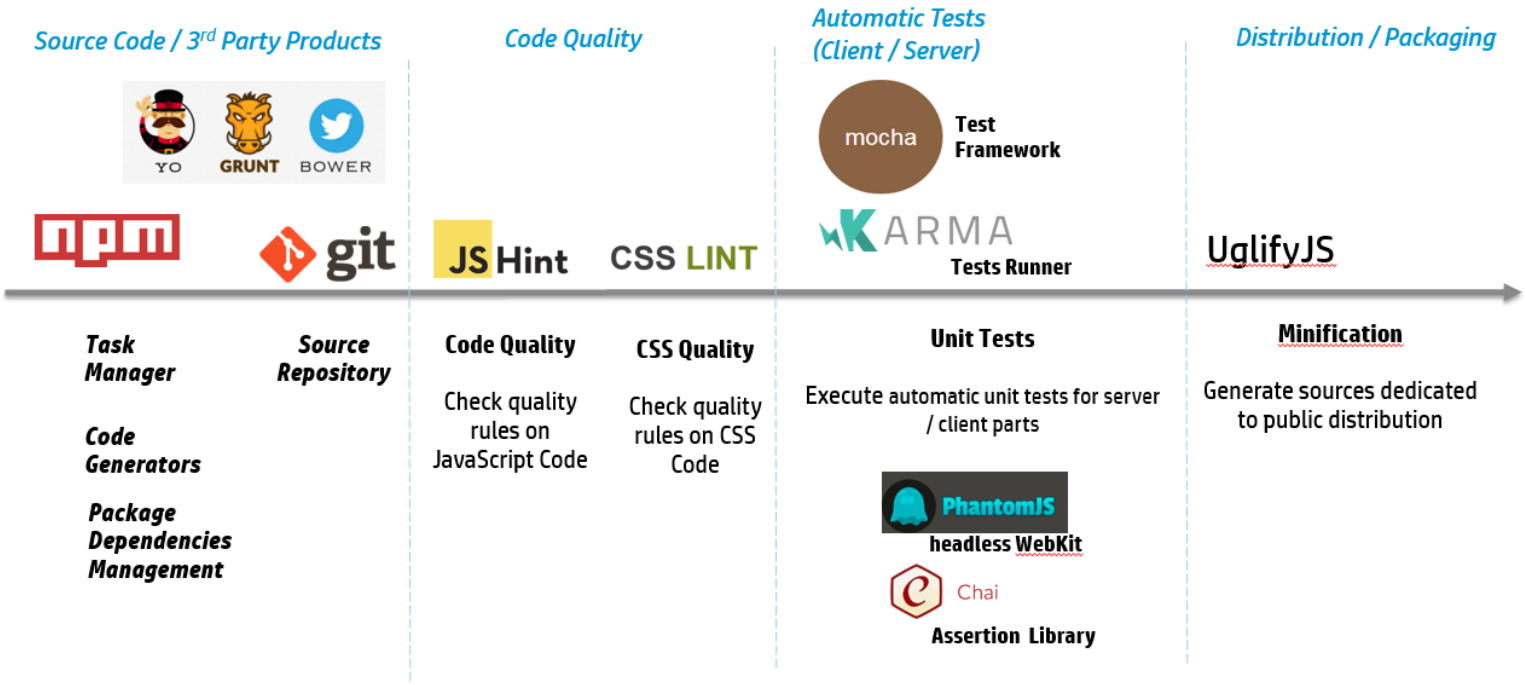


Figure 16: Development cycle and tools

4.2 Add-on ID Registration

To create a new Add-on for Unified Console V2.x and to make sure this add-on will be compatible with others installed on the same platform, a registration process has been put in place to get a unique identifier to be used (as prefix or as a directory name) in all components you will develop.

IMPORTANT: This is mandatory for Add-on you need to deliver in production. This identifier will be unique and validated by the HPE Support Team (https://softwaresupport.hp.com/)

It will be a show-stopper problem on a production platform, if there is a conflict in Add-on identifier.

Example of Identifier already attributed:

ID	Description
HP	HP Unified Console (before November 2015)
HPE	HPE Unified Console (Core)
OSSA	OSS Analytics
OSSM	OSS Monitoring
NFVD	NFV Director
NFVA	NFV Analytics
FAS	Fault Analytics and Statistics
TRAINING	Training Team

4.3 Setup Add-on Project

To easily setup a new add-on project for Unified Console, you can use the SDK kit provided.

1. Make sure you have installed all the needed development tools (check 3.2 Install Development Tools)
2. Create a development directory for your add-on like c:\UOC_Addon
3. Download the SDK kit for the exact same version as your UOC runtime. (UOCV2.3.0-MR-SDK.zip)
4. Unzip this archive in your development directory
5. Init the GUI Database (CouchDB)

```
grunt db:init
```

This command empty the database and populate it again with the content of the Data local directory (like c:\DEV_UOC\Data) and the Core Data local directory (like c:\DEV_UOC\Core\Data)

6. Test the installation starting UOC server and connect a web browser
   a. Open a command line in your development directory
   b. Start UOC with the command 'grunt' as administrator

```
grunt
```

IMPORTANT: The administrator right is needed because 'grunt' will create dynamically links between core code and your add-on development to allow you to start UOC directly from your add-on and avoid any replication of components to another directory. Every directories in your add-on will be linked to the similar directories in Core directory. So, there is a clear split between your own code and the UOC core foundation but granting you have a live and rapid experience.

The development SDK kit cannot be used in production. This kit has been tuned especially for development. It means most of caches are disabled and optimization usually done for runtime kit have been skipped to ease development, and troubleshooting.

7. Start a Web browser and enter URL: http://localhost:9000
8. The UOC login page appears. You can log as admin/admin
9. UOC Development environment is ready for your add-on development.

4.4 Add-on Project Overview

The Add-on Development kit create the following directories, Main directories are Client, Server, Core and Data. Here after the detailed information about the content of the SDK.

Files and Directories	Description
Client	Client part of the add-on (leverage AngularJS Framework v1.x)
• addons	Directory where to develop the client components of the add-on: layouts, menu-bars, menu-items, modules, widgets, launch-keywords and themes.
• bower_components	List of all 3rd party products installed for the client part (installed using bower install)
• public	Public client directory to store resources needed at the client side. Note: This directory will not be uninstalled on a runtime kit.
o images	Images and icons. All is PNG format.
▪ launch-keywords	Images for launch keywords. It will be displayed in the Add-on Management pages.
▪ layouts	Images for layout. It will be displayed in the Add-on Management pages.
▪ menu-bars	Images for menu bars. It will be displayed in the Add-on Management pages.
▪ menu-items	Images for menu items. It will be displayed in the Add-on Management pages.
▪ modules	Images for modules. It will be displayed in the Add-on Management pages.
▪ notifications	Images to associated to real-time notifications.
▪ packages	Images for packages. It will be displayed in the Add-on Management pages.
▪ themes	Images for themes. It will be displayed in the Add-on Management pages.
▪ widgets	Images for widgets. It will be displayed in the Add-on Management pages.
▪ workspaces	Images for workplaces. It will be displayed in the Add-on Management pages.
o maps	GeoJSON map definition files. See https://tools.ietf.org/html/rfc7946
o sounds	MP3 notification sound files
Core	Reserved directory for the UOC core foundation. DO NOT CHANGE ANYTHING. This part should strictly match the UOC runtime kit.
Data	Local data directories Note: This directory will not be uninstalled on a runtime kit.
• categories	JSON File definition for workspace categories
• launch-categories	JSON File definition for launch categories

• launches	JSON File definition for launches
• notifications	JSON File definition for notifications
• permissions	JSON File definition for permissions
• roles	JSON File definition for roles
• states	JSON File definition for advanced lifecycle of UOC object.
• users	JSON File definition for user (local authentication)
• views	JSON File definition for views
• workspaces	JSON File definition for workspaces
Logs	Logs directories
Node_modules	List of all 3rd party products installed for the client part (installed using npm install)
Server	Server part of the add-on (leverage NodeJS technologies)
• addons	Directory where to develop the server components of the add-on: Plugins
• public	Public server directory to store resources needed at the server side. Note: This directory will not be uninstalled on a runtime kit.
o addons	Server add-ons directory
▪ plugins	Server add-ons : Plugin directory
.bowerrc	Bower Configuration File (proxy setting) See https://bower.io/docs/config/
.csslintrc	CSS Lint Configuration File (options and global settings). See https://github.com/gruntjs/grunt-contrib-csslint
.jshintrc	JS Hint Configuration File (options) See http://jshint.com/docs/
bower.json	Bower Packages definition to list all the 3rd party products needed for the client part. See https://bower.io/docs/creating-packages/
core-main.js	UOC Core main dependencies. RequireJS definition for core components and 3rd Party Product. Note: You can see all the alias used for require dependencies injection.
Gruntfile.js	Grunt Configuration File. See http://gruntjs.com/
karma.conf.js	Karma Configuration File (options) See http://karma-runner.github.io/1.0/config/configuration-file.html
package.json	NPM Package Definition to list all the 3rd party products needed for the server part. See https://docs.npmjs.com/files/package.json
README.txt	Readme File

test-main.js	Main configuration file for automatic tests (karma, mocha, chai). This file is ready to use, do not change anything unless you are fully confident about your changes.

The add-on developer usually develops his components in server for its plugins, and client for the graphical client components. Local data is all the definition that will be used to populate the GUI database.

IMPORTANT: Never modify the Core directory that contains all the UOC Core component use to provide the same level of feature as the runtime kit.

4.5 Add-on Project Grunt Tasks

IMPORTANT: All command line window must run with administrator privileges.

4.5.1 Run a Command line as Administrator

The SDK and the task manager will need administrator right to execute properly and be able to create dynamic link between the core directory and your own add-on directory.

This is mandatory to have a seamless integration between UOC and your custom add-on and behave exactly like on a runtime platform when you install your add-on on top of the UOC kit.

If you start a command line without this administrator rights. You will notice warning at the beginning of the UOC start and all your add-on components will not be available in UOC.

To start a command line window with administrator privileges, you can check the documentation of your windows version but here after the most common ways.

4.5.1.1 Create a Shortcut for Command Prompt on the Desktop

Right-click an empty spot on the Desktop. From the context menu, select “New > Shortcut”.

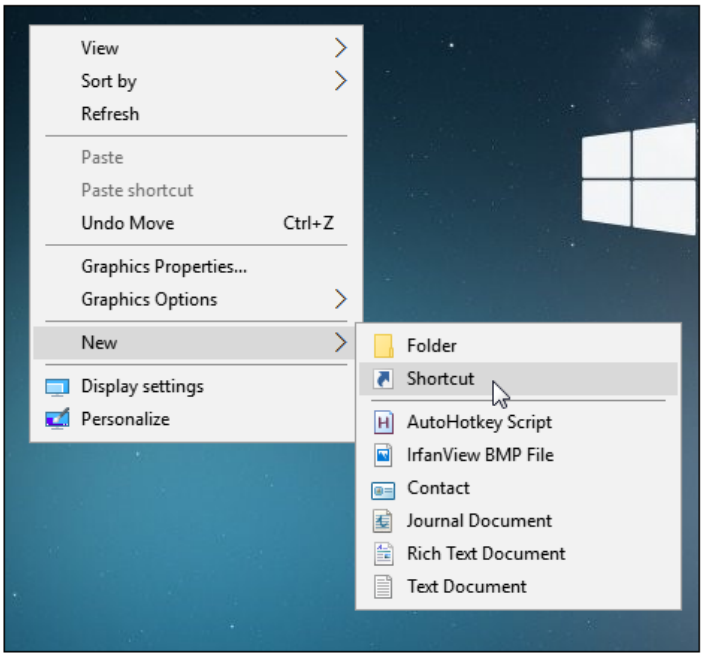


Figure 17: Create a Shortcut for Command Prompt on the Desktop (Windows 10)

In the box labeled “Type the location of the item,” enter “cmd.exe”.

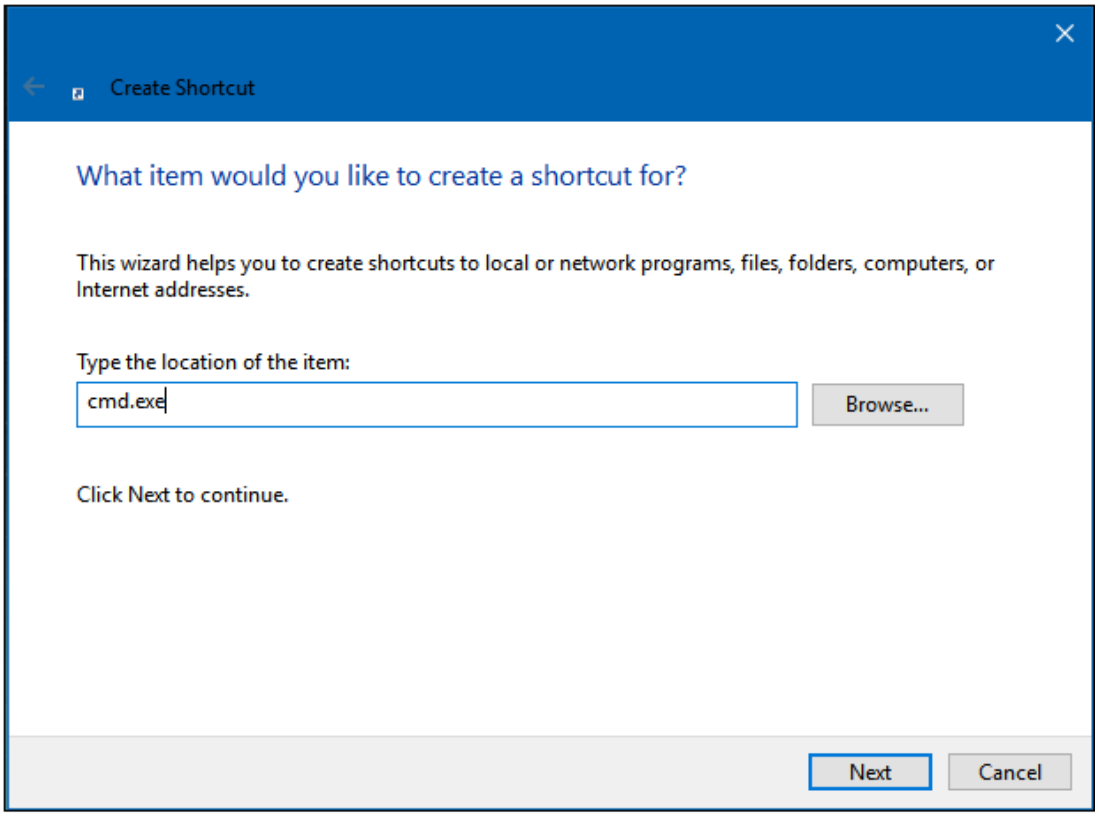


Figure 18: Create a Shortcut for Command Prompt on the Desktop #2 (Windows 10)

Press “Next”, give the shortcut a name and choose “Finish”.

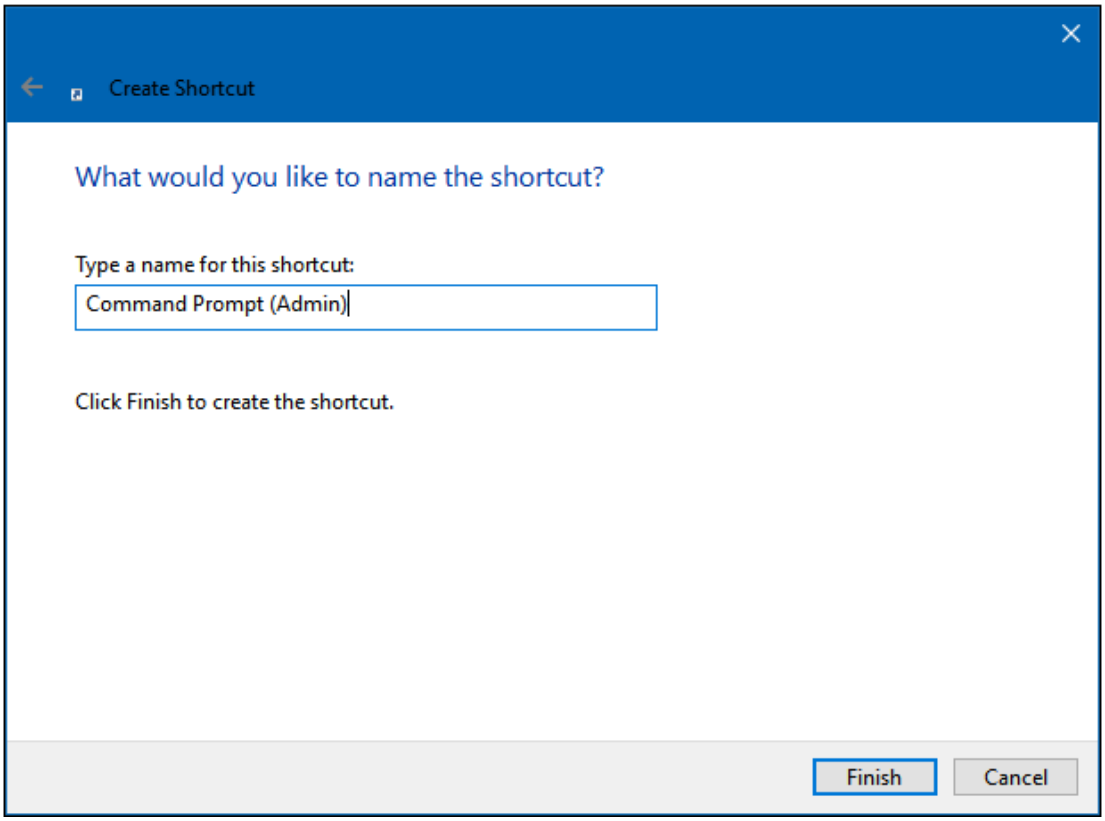


Figure 19: Create a Shortcut for Command Prompt on the Desktop #3 (Windows 10)

If you want to open the command prompt in Administrator mode, right-click on the new shortcut icon and choose “Properties” from the context menu. Click the “Advanced” button and check “Run as administrator”.

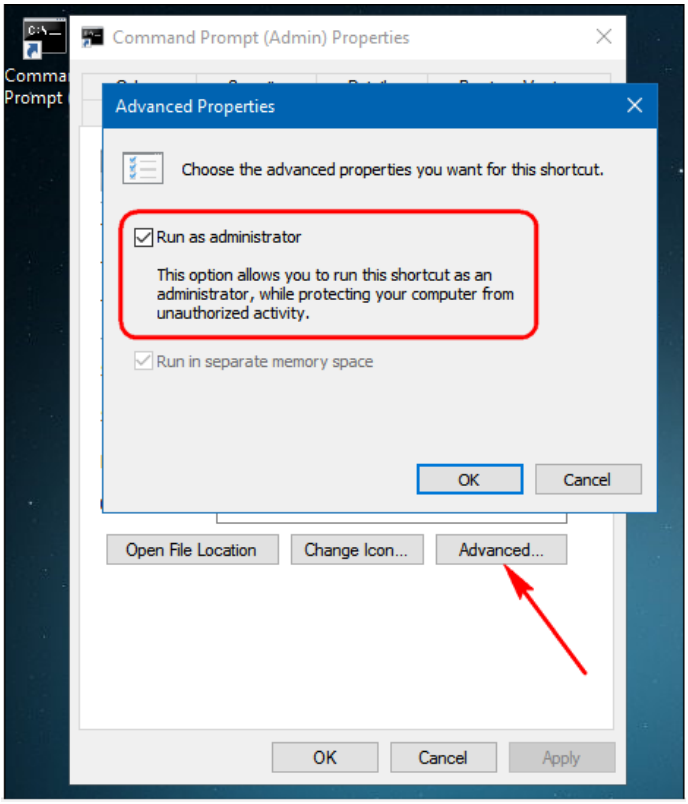


Figure 20: Create a Shortcut for Command Prompt on the Desktop #4 (Windows 10)

Now you have created a shortcut that will open the command prompt with administrative privileges when double-clicked.

4.5.1.2 Open Command Prompt from Search

You can easily open the command prompt by typing “cmd” into the search box (Win + S). Alternatively, click/tap on the microphone icon in Cortana’s search field and say “Launch Command Prompt”.

To open as Administrator, type cmd into the search box, and either right-click and choose Run as Administrator, or highlight the result with the arrow keys and press CTRL + SHIFT + ENTER to open an administrator mode command prompt.

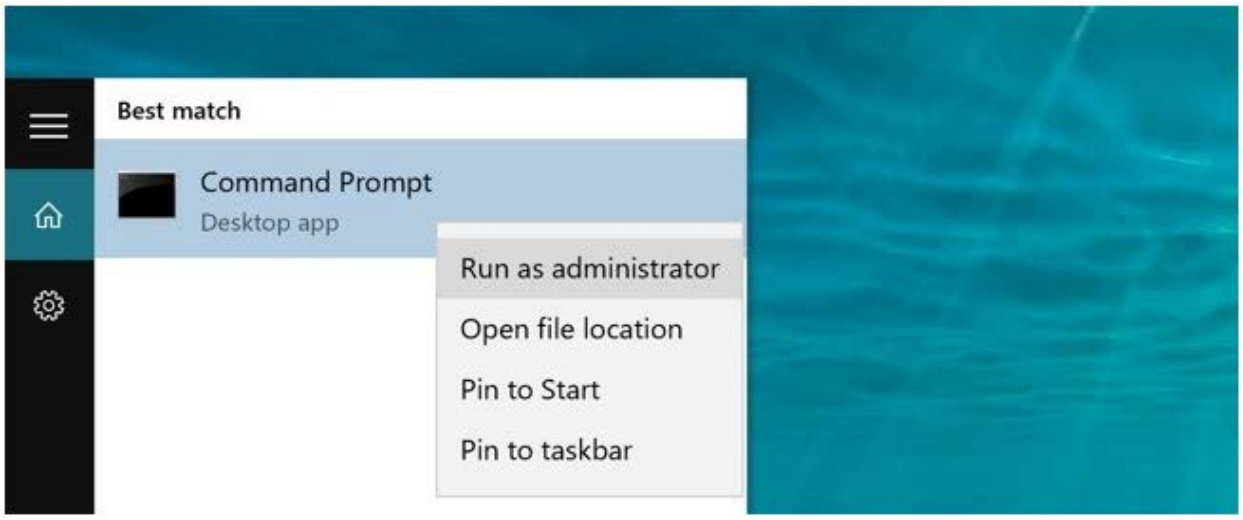


Figure 21: Open Command Prompt from search (Windows 10)

4.5.1.3 Open Command Prompt from Win+X Power Menu

Press “Win + X”, and click/tap on Command Prompt, or Command Prompt (Admin) to open it in Administrator mode

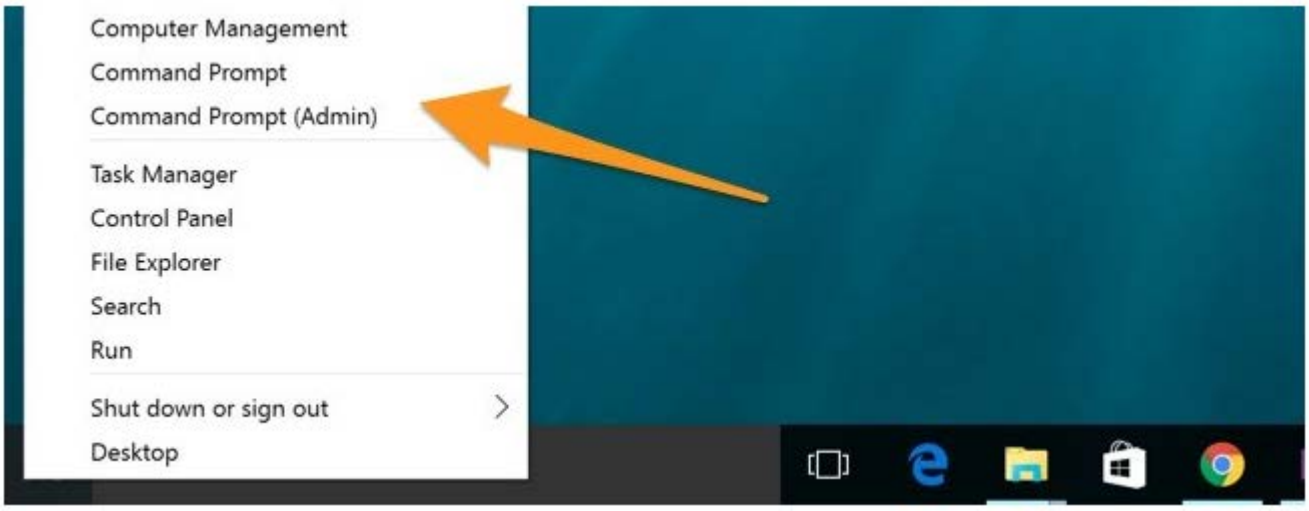


Figure 22: Open Command Prompt from Win+X Power menu (Windows 10)

4.5.1.4 Open Command Prompt from Task Manager

Open Task Manager with more details. Choose File and then Run New Task. Type cmd or cmd.exe, and hit OK to open up a regular command prompt. You can also check the box to open as administrator.

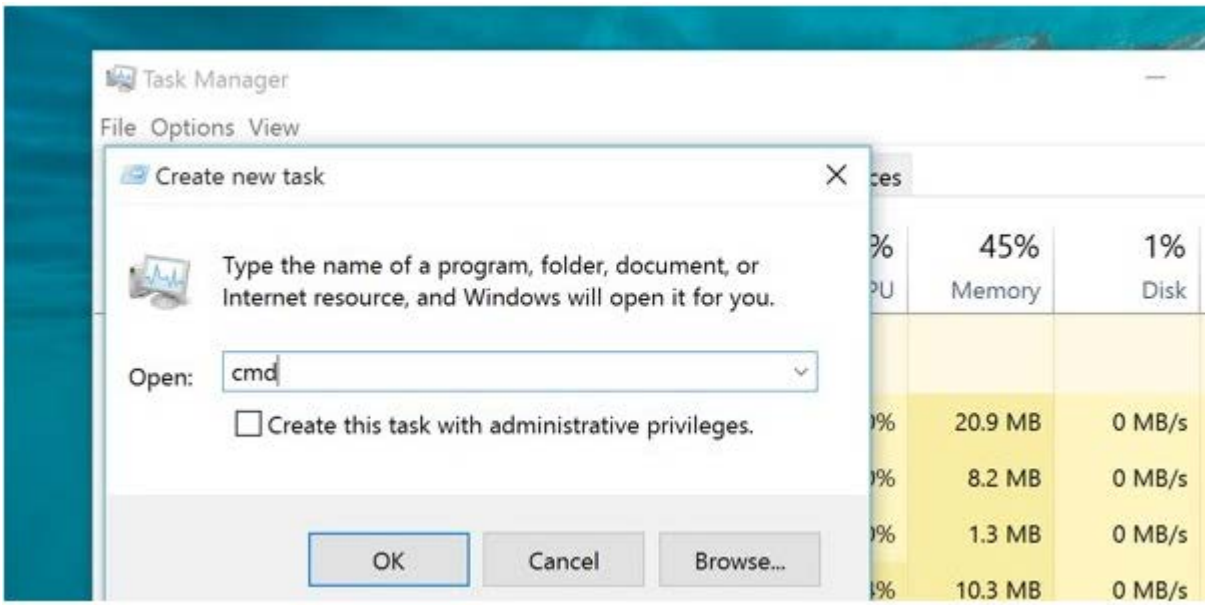


Figure 23: Open Command Prompt fromTask Manager (Windows 10)

4.5.1.5 Open a Command Prompt in Admin Mode from Task Manager the Secret Easy Way

To open a command prompt in Administrator mode from the Task Manager, you can hold the CTRL key while clicking File -> Run New Task and it'll immediately open an administrator mode command prompt.

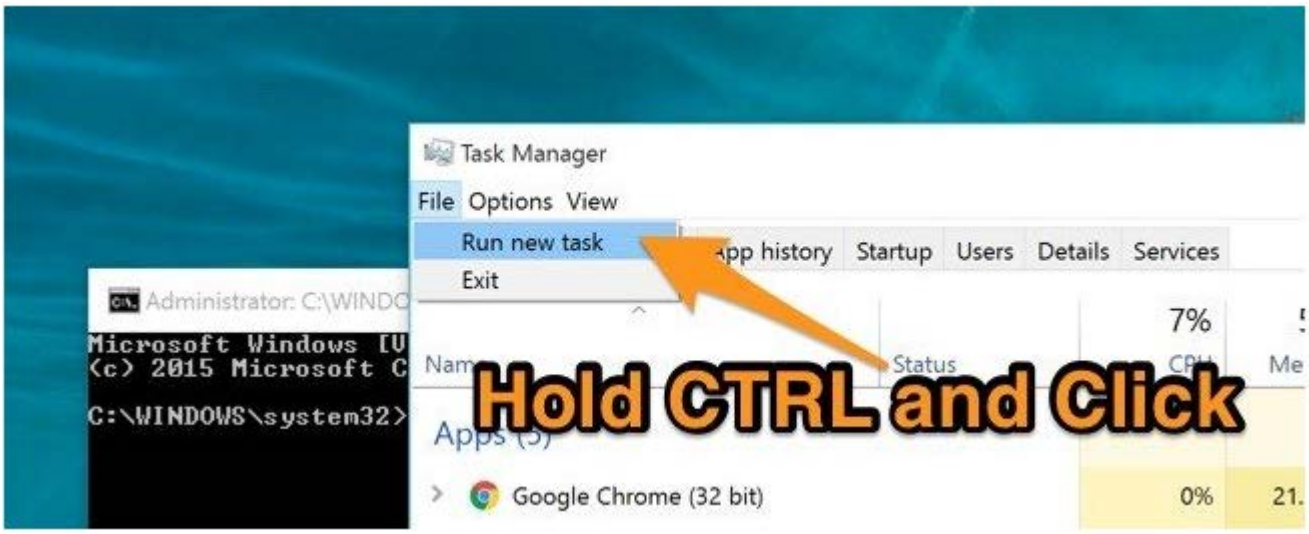


Figure 24: Open a Command Prompt in Admin Mode from Task Manager the Secret Easy Way (Windows 10)

4.5.1.6 Open Command Prompt from All Apps in Start Menu

Open the Start menu, and click/tap on All apps at the bottom.

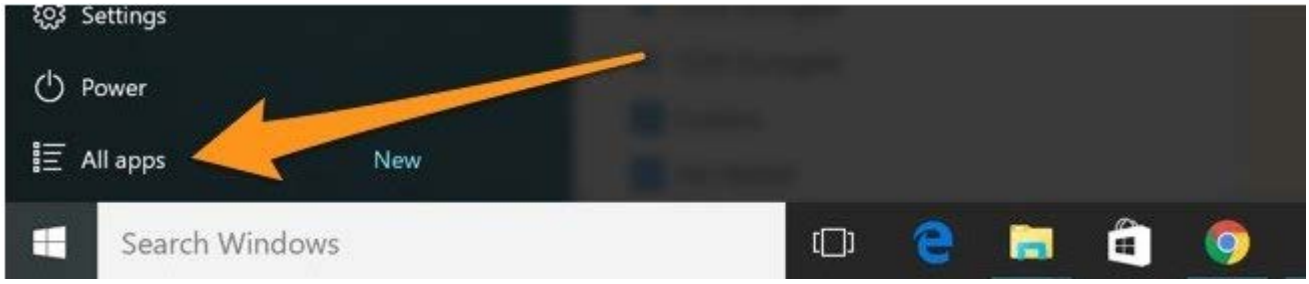


Figure 25: Open Command Prompt from All Apps in Start Menu (Windows 10)

In All apps, scroll down and expand the Windows System folder, then click/tap on Command Prompt.

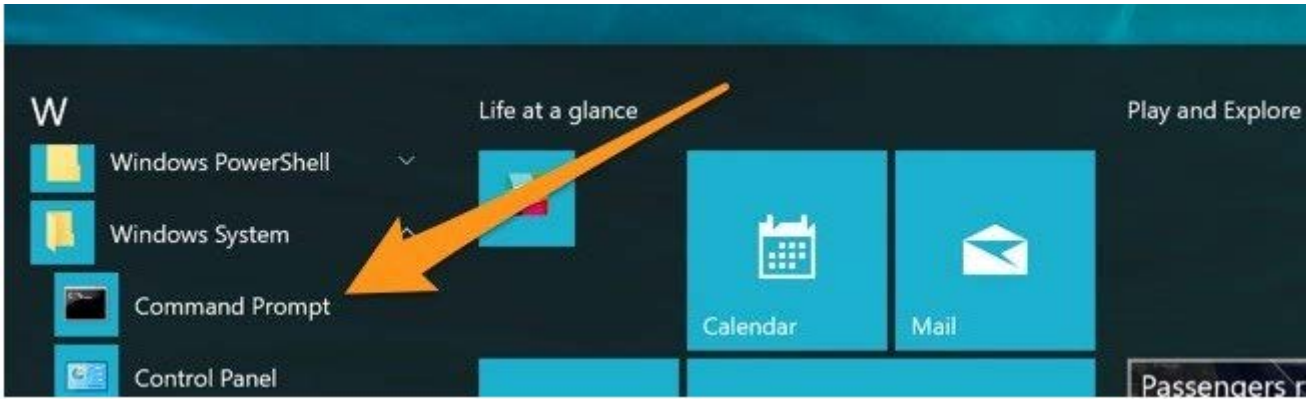


Figure 26: Open Command Prompt from All Apps in Start Menu – Windows System (Windows 10)

4.5.1.7 Open Command Prompt from File Explorer

Open File Explorer, and navigate to the C:\Windows\System32 folder, and click/tap on cmd.exe. You can actually do this from any file browser window by right-clicking on cmd.exe and choosing Open.

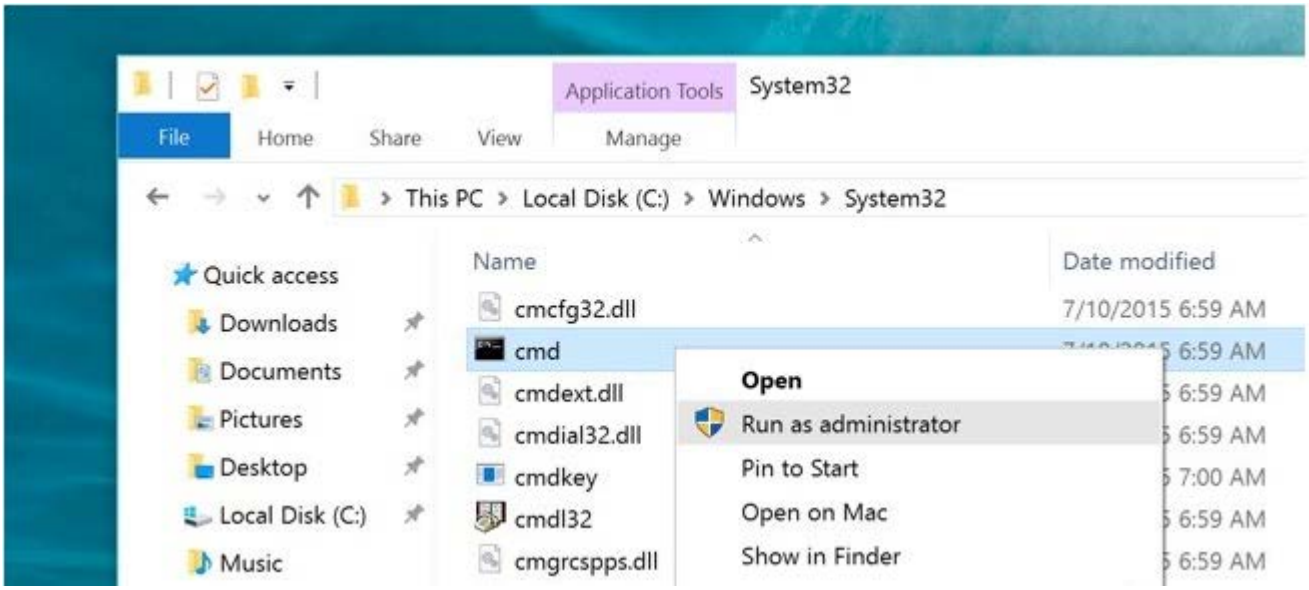


Figure 27: Open Command Prompt from File Explorer (Windows 10)

4.5.1.8 Open Command Prompt Here from File Menu

Open File Explorer, select or open a folder or drive where you want to open the command prompt from. Click/tap on the "File" tab on the Ribbon, and select "Open command prompt". It has two options:

- Open command prompt — Opens a Command Prompt within the currently selected folder with standard permissions.
- Open command prompt as administrator — Opens a Command Prompt within the currently selected folder with administrator permissions.

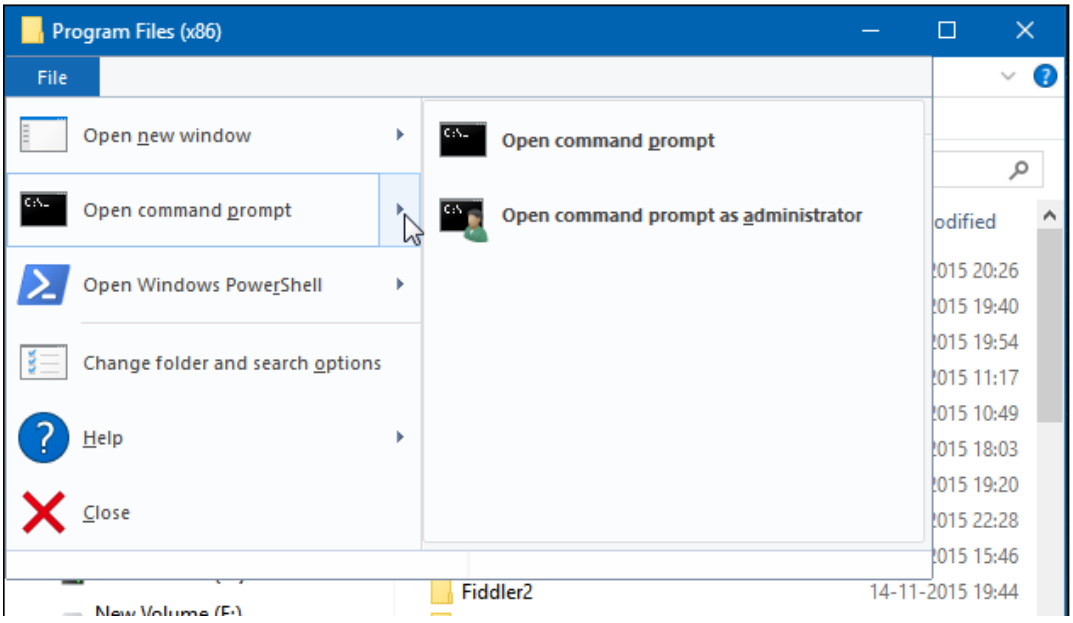


Figure 28: Open Command Prompt Here from File Menu (Windows 10)

4.5.2 Task Manager Usage

You can use the following command to get details about available tasks in the UOC SDK.

```
grunt –-help
```

or

```
grunt -h
```

Here are the public tasks available during your UOC Add-on development:

IMPORTANT: All other tasks prefixed by '*' (wildcard) is internal grunt sub-tasks.
Please do not use them

Tasks	Description
grunt grunt serve	Start Unified Console in command-line in development mode with live reload.
grunt serve:production grunt default	Start Unified Console in command-line in production mode with live reload.
grunt –help | -h	Display the available tasks usage
grunt jshint	Start Static code analysis for client and server code and display results
grunt jshint:client	Start Static code analysis for client code and display results
grunt jshint:server	Start Static code analysis for server code and display results
grunt csslint:client	Start CSS Analysis and displays results
grunt db:init	Reinit the GUI database (CouchDB). Reset and populate schema and data. Note: This task can take option parameters if the database admin's credential are different from the default (admin:admin) grunt db:init --adminUsername [adminUsername] --adminPassword [adminPassword]
grunt sdk:version	Display the UOC SDK version (e.g. 2.3.0-MR)
grunt test	Run client and server automatic tests written by the UOC add-on developer.
grunt test:client	Run client automatic unit tests written by the UOC add-on developer.
grunt test:server	Run server automatic unit tests written by the UOC add-on developer.
grunt build:kit	Build a runtime add-on distribution kit. It creates a 'dist' directory with all minified sources. This directory is ready to be tested and packaged (additionally with external tool produce a RPM, zip, etc.)
grunt generate	

Figure 29: Task Manager Usage (grunt --help)

4.5.3 Start UOC

By default, UOC starts in development mode. i.e. without any cache enabled and using the connect live reload feature to ease the automatic reload of the web browser in case of code changes.

It is possible to force a specific mode of NodeJS mode defining this environment variable.

```
SET NODE_ENV=production | development
```

To start UOC Server, you can select one of this command. You need to start a command line with administrator privileges. See 4.5.1 Run a Command line as Administrator.

```
grunt
grunt serve
grunt default
grunt serve:production (to force a production mode)
```

A console logs several useful information:

- GUI database information (protocol, port, version)
- List of loaded plugins and their directories
- OS name, Type, platform
- Server protocol, port
- Timeout of http request (default is unlimited).
- Server body parser (http POST configuration). Default is 100KB
- Node directory and Mode (development or production)
- Authentication mode (local or SAML), Default is local.
- Startup parameters to drive the import policy in the GUI database.
- Starting time of the server

TIP: To stop UOC, you need to use Ctrl-C or Ctrl-Break or kill the commandline window.

4.5.4 Analysis Static JavaScript Code

As an Add-on developer, it is recommended to always check if your code raised some issue analyzing the code. JSHint is a very powerful javaScript tool that you can easily customize to add / remove or tune rule checking.

See http://jshint.com/docs/options/

```
grunt jshint
```

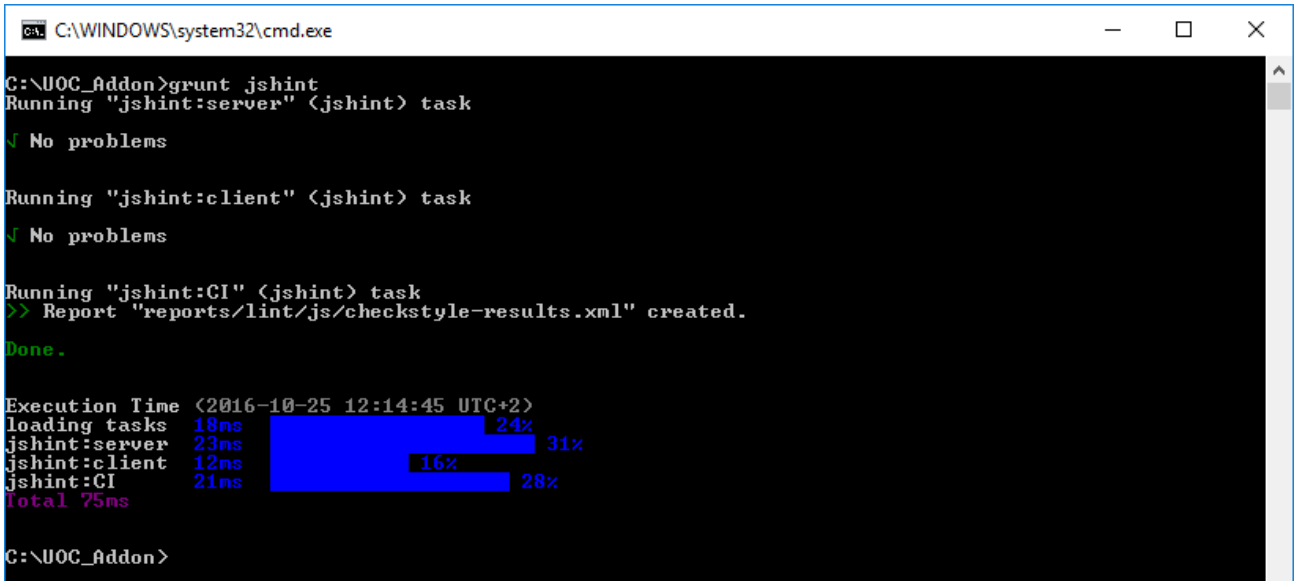


Figure 30: Static Code Analysis (grunt jshint)

4.5.5 Analysis CSS files

As an web designer, it is recommended to always check if your CSS raised some issue. CSSLint is a very powerful CSS tool that you can easily customize to add / remove or tune rule checking.

See https://github.com/CSSLint/csslint/wiki

```
grunt csslint:client
```

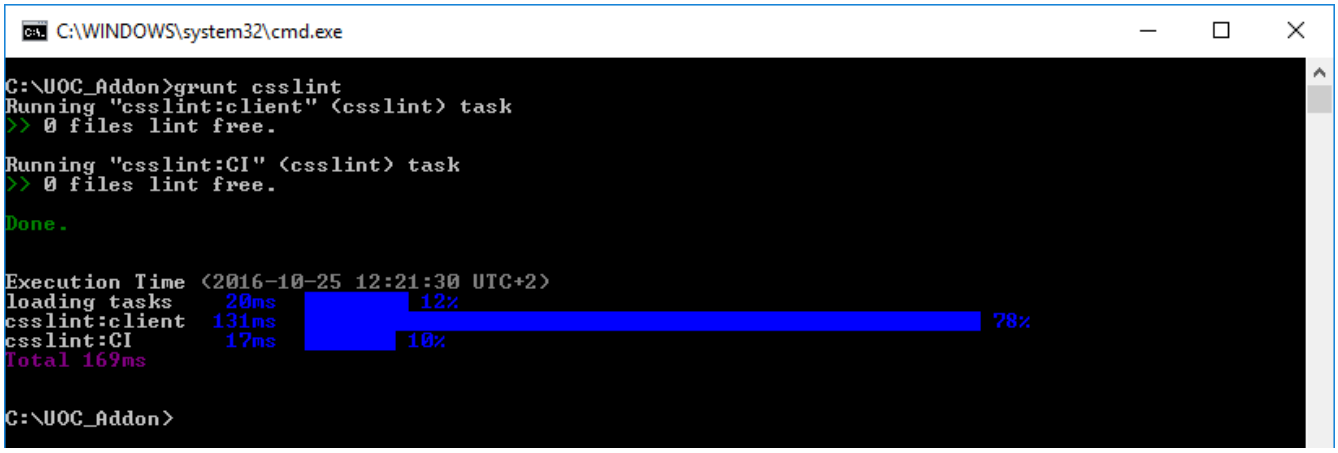


Figure 31: Static CSS Analysis (grunt csslint)

4.5.6 Run Automatic Unit Tests

It is recommended to leverage the Test Framework embedded in UOC SDK (Mocha / Karma…) to setup automatic tests to avoid any regression of your code. Note that during an initial generation of your component, you have the opportunity to generate skeleton for tests.

Running tests is very easy and you can run these following commands.

```
grunt test
grunt test:server
grunt test:client
```

The test framework will dynamically explore your add-on to find on your client, server or both all test files, e.g. the files with the suffix: *-test.js (hpe-meter-directives-test.js, hpe-meter-service-test.js, package-restapi-test.js, etc.). The test file can apply on any angular components (directives, controllers, services, filters, modules…) for the client side and any JavaScript module for the server side.

UOC recommend to keep these test files included in each module to ease the design of large add-on and create unnecessary dependencies between directories.

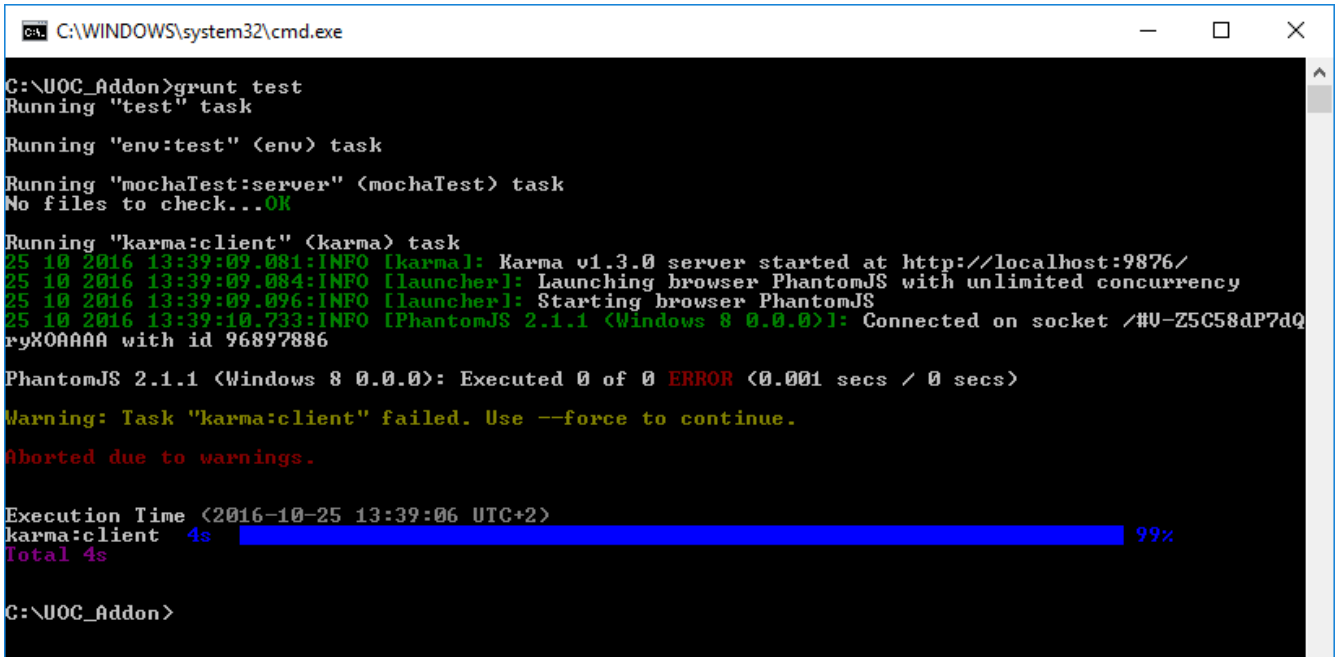


Figure 32: Automatic client and server unit tests (grunt test)

4.5.7 Build Distribution Kit

When your add-on development is over, you will need to package your add-on to be installed on top of the Unified Console runtime kit, the SDK provides you the first step of the kitting. It prepares all the files in a dedicated directory named 'dist' in your add-on directory to give you the distribution files you need to package.

NOTE: What is minification ?

Minification in JavaScript is the process of removing all characters that are not necessary from the JavaScript source code without impacting the functionalities of the code. Code that has gone through the minification process is also known as "minified" .
The data that is removed in JavaScript minification is extra whitespace, comments, new line characters, etc.

What are the benefits and advantages of JavaScript minification?

The minifcation is not mandatory but the main purpose is to speed up the downloading or transfer of the code from the server hosting the website's JavaScript and also minimize the memory footprint. It is a recommended best practice from the web standard.

To prepare a packaging, you can use the following command.

```
grunt build:kit
```

The build process uses several other tools to clean directories, annotate code, minifiy (css, html, js) and copy results in the dist directory.

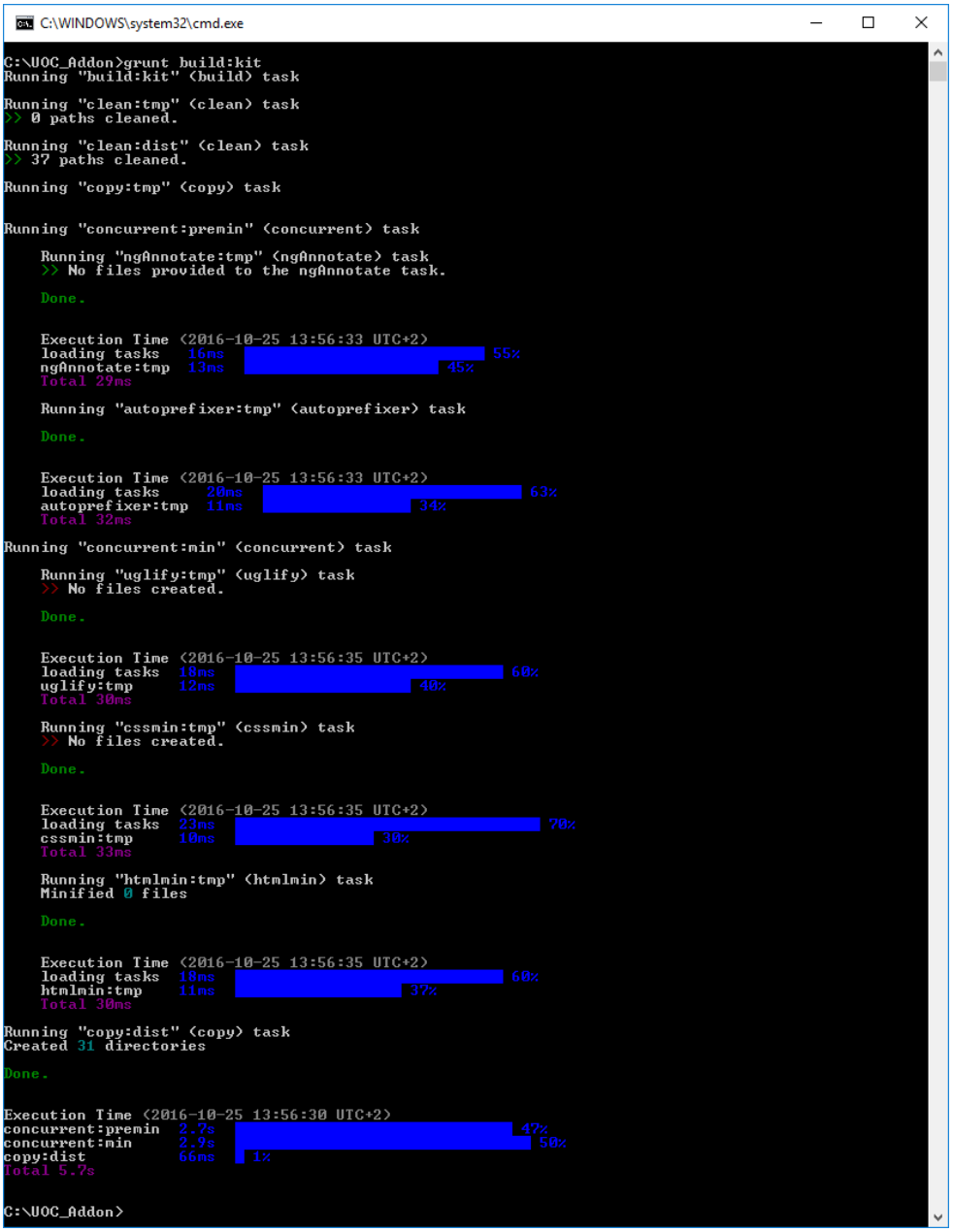


Figure 33: Build distribution directory (grunt build:kit)

In your distribution directory (dist), you will see the result of the minification and the main parts: server, client and data you need to install on top of the UOC runtime kit.

Files and Directories	Description
Client	Client part of the add-on (leverage AngularJS Framework v1.x)
• addons	Directory where to develop the client components of the add-on: layouts, menu-bars, menu-items, modules, widgets, launch-keywords and themes.
• public	Public client directory to store resources needed at the client side.
Data	Local data directories
Server	Server part of the add-on (leverage NodeJS technologies)
• addons	Directory where to develop the server components of the add-on: Plugins
• public	Public server directory to store resources needed at the server side.

It is up to the add-on developer to build on top of the build distribution, an additional process to produce a kitting format ready to install. It is recommended to add some Linux script to package as a RPM the add-on.

TIP: UOC Core build its own RPM kit leveraging the open source **easy-rpm.** Check their website https://github.com/panitw/easy-rpm for details

4.5.8 Init GUI Database

You can use the following command to initialize or reset the GUI database (CouchDB).

IMPORTANT: This command will delete all the data stored in the GUI Database, recreate all the schema and populate the database using the Data directories (add-on and core).
If you need to keep existing document from the database, you need to export yourself the JSON document

```
grunt db:init
```

If your GUI Database admin credentials are different from the default one (admin:admin), you need to enter your credentials with the following options:

```
grunt db:init --adminUsername=bob --adminPassword=bob
```

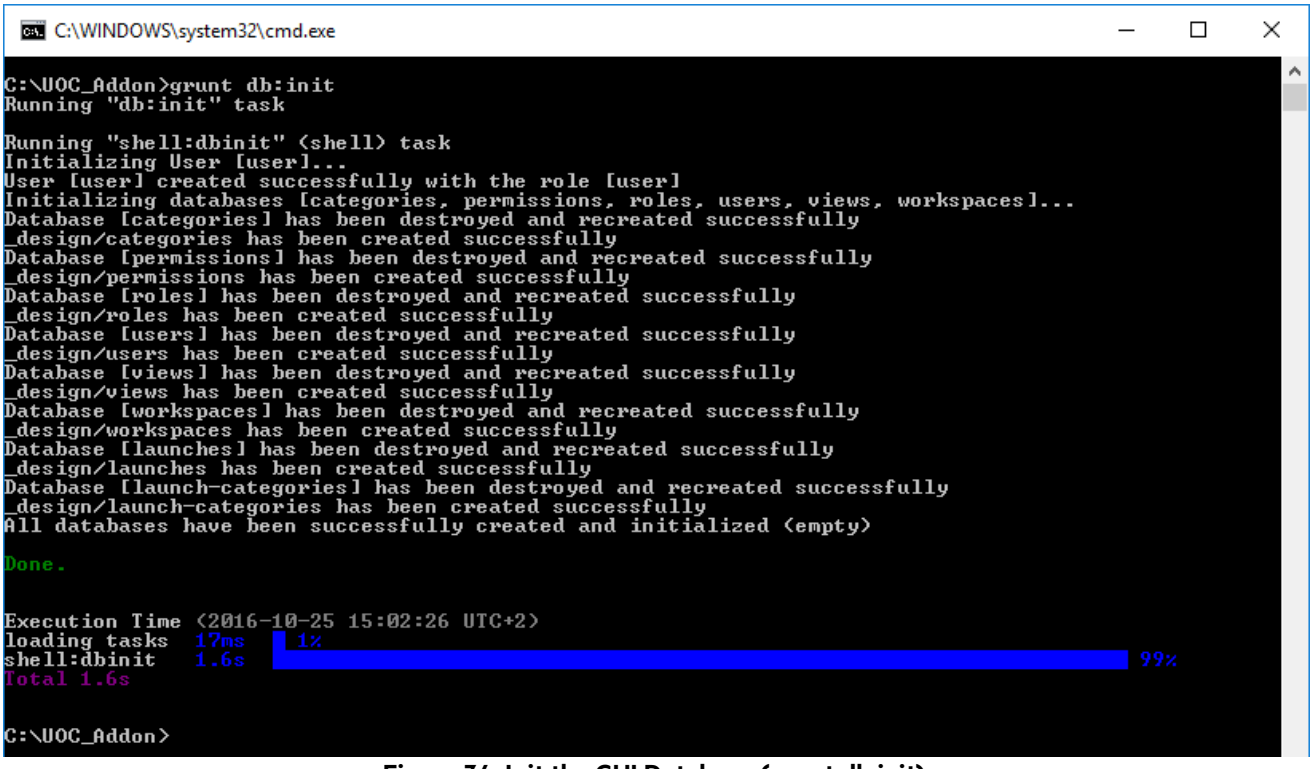


Figure 34: Init the GUI Database (grunt db:init)

4.5.9 Use Add-on Generators

The SDK integrates code generator to scaffold quickly a working skeleton for UOC. You can use the following command to start generators.

```
grunt generate
```

The first time you will use the generator, it will ask you to complete a set of question. These answers will be saved and reused for your future generation. See 4.2

Add-on Profile for generators

The next time you start the generator, it will provide you a menu selection to generate your component. It is also possible to directly generate a specific component specifying the component identifier in the command line.

Please read the 4 Add-on Generators for a more detailed description and usage.

Chapter 5
Add-on Generators

IMPORTANT: To make sure that all extension made by all can run on the same UOC platform, strict naming rules apply. They are based on the Add-on Identifier as prefix and the identifier of the component. This unique identifier is used for all naming conventions of files.

Make sure you follow these naming rules to avoid any conflict or trouble that can be in some case show-stopper if you want to install your own add-on with many others on the same UOC platform.

TIP: For Windows developers: generators prompting system does not work properly with Git Bash. It is recommanded to use Windows command prompt or Powershell.

5.1 Overview

The SDK integrates code generators to scaffold quickly a working skeleton for UOC. You can use the following command to start generators.

```
grunt generate
```

The first time you will use the generator, it will ask you to complete a set of question. These answers will be saved and reused for your future generation. See 5.2 Add-on Profile for generators.

The next time you start the generator, it will provide you a menu selection to generate your component.

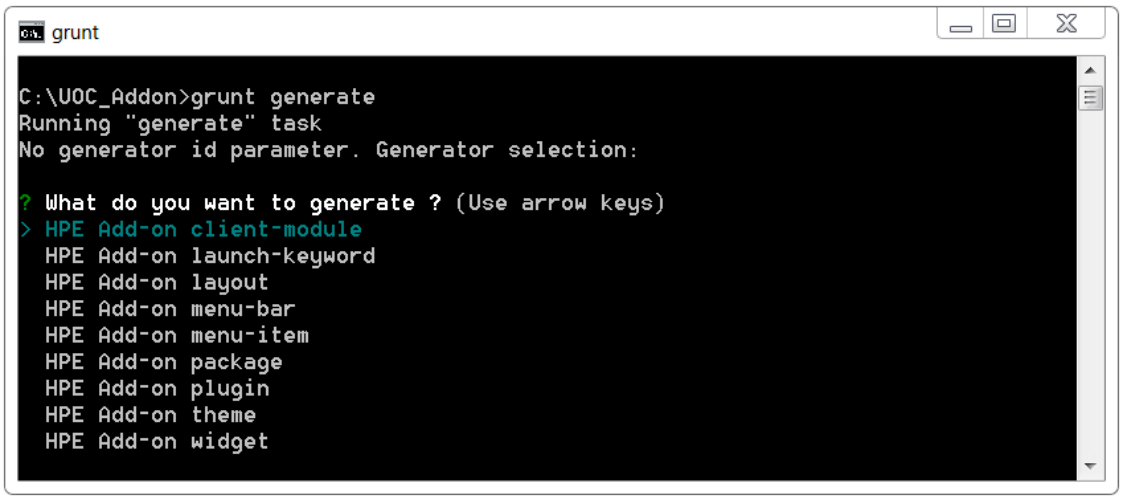


Figure 35: Generator – Menu selection for generators

It is possible to directly generate a specific component specifying the component identifier in the command line:

```
grunt generate:client-module
grunt generate:launch-keyword
grunt generate:layout
grunt generate:menu-bar
grunt generate:menu-item
grunt generate:package
grunt generate:plugin
```

```
grunt generate:theme
grunt generate:widget
```

5.2 Add-on Profile for generators

1. What is your Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration). It must be in lowercase and with no special character.
2. Author of the add-on ?
3. Default version to use ? (default is 1.0)

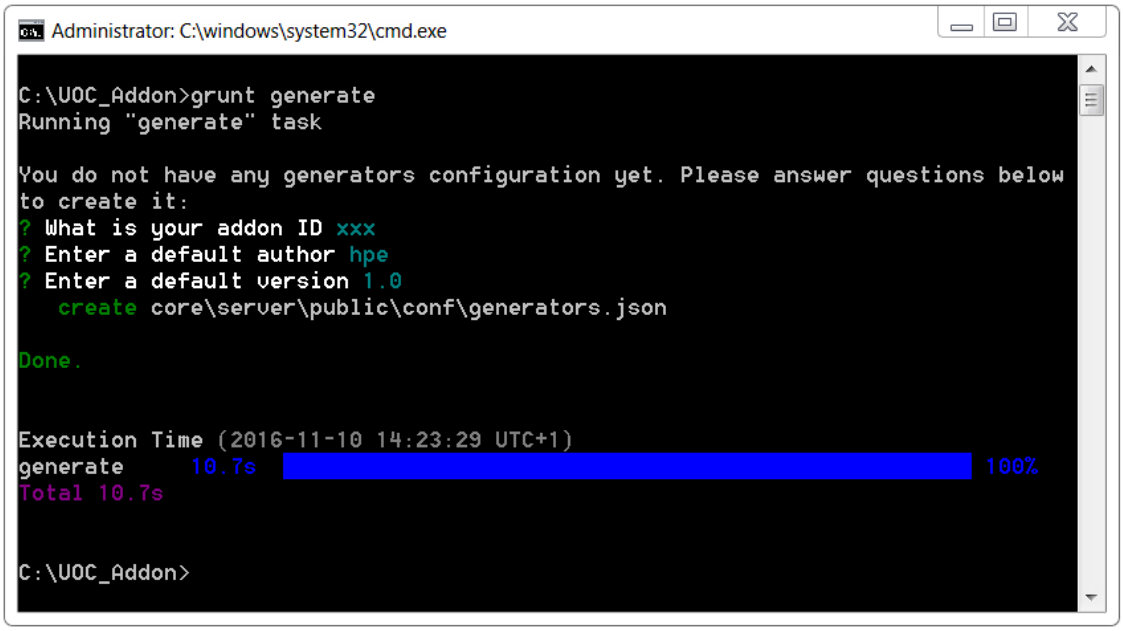


Figure 36: Generator - Create your add-on profile

Information are saved in the <install_dir>\core\server\public\conf\generators.json

```
{
        "defaults": {
                "addonId": "xxx",
                "author": "hpe",
                "version": "1.0"
        }
}
```

TIP: If you need to change this add-on profile. You can use the command grunt generate:generators-config or delete the generated file and go through 'grunt genererate' to start the wizard or manually change the information in the JSON generated file.

5.3 How to generate an add-on client Module

There are several types of module that can extend the UOC platform:

- Generic (to provide an universal skeleton for any kind of module)
- Filter (to filter or define new display formatter)
- Action (to define new action services)

The generator will adapt its questions to the type of module you need.

5.3.1 How to generate an add-on Module Generic

You can use the following command to generate a module component.

```
grunt generate
```

Then select in the menu selection: **HPE Add-on client-module** -or- enter directly the following command.

```
grunt generate:client-module
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration) *Format: Only letters and "-" are allowed. It must start and end with a letter.*
2. Enter the module identifier. This id will be prefixed by the add-on id in the generated file (separated by a "-"). *Format: Only letters and "-" are allowed. It must start and end with a letter.*
3. Enter the module description
4. Enter the module author or validate the default one you previously entered in your profile
5. Enter the module version or validate the default one you previously entered in your profile
6. Specify '**generic**' in the type of module you want to generate. For this type, you need to cover multiples need, so the generator asks you all the files it can generate for your module.
7. Confirm if you want to generate a directives file ?
8. Confirm if you want to generate a controllers file ?
9. Confirm if you want to generate a services file ?
10. Confirm if you want to generate a filters file ?
11. Select if you want to generate skeleton files for automatic tests
12. Select localization files to generate (all installed language codes by default).

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Module will be placed under <install_dir>\client\addons\<my add-on Id>\modules\<my add-on id>-<my module id> directory and will be composed of the following files:

Files	Description
modules.json	Descriptor of the module
<my add-on id>-<my module id>.css	CSS file containing specific style of the graphical part of the module (if directives were generated)
<my add-on id>-<my module id>.html	HTML file containing HTML markup of the graphical part of the module (if directives were generated)
<my add-on id>-<my module id>.js	Implementation of the main module for the module
<my add-on id>-<my module id>-controllers.js	Implementation of controllers for the module (if controllers were generated)
<my add-on id>-<my module id>-controllers-test.js	Controllers unit test file validating that controllers can be loaded without error. (if controllers were generated)
<my add-on id>-<my module id>-directives.js	Implementation of directives for the module (if directives were generated)
<my add-on id>-<my module id>-directives-test.js	Directives unit test file validating that directives can be loaded without error. (if directives were generated)

<my add-on id>-<my module id>-services.js	Implementation of services for the module (if services were generated)
<my add-on id>-<my module id>-services-test.js	Services unit test file validating that services can be loaded without error. (if services were generated)
<my add-on id>-<my module id>-filters.js	Implementation of filters for the module (if filters were generated)
<language code>.json	One localization JSON file per selected supported language codes. *(e.g. en-us.json, fr-fr.json,...)*

NOTE: You can customize a specific module icon in:
<install_dir>\client\public\images\modules\<my add-on id>-<my-module id>.png

Default icon is under <install_dir>\client\public\images\modules\default.png

It is also recommended to review and adjust the descriptor.

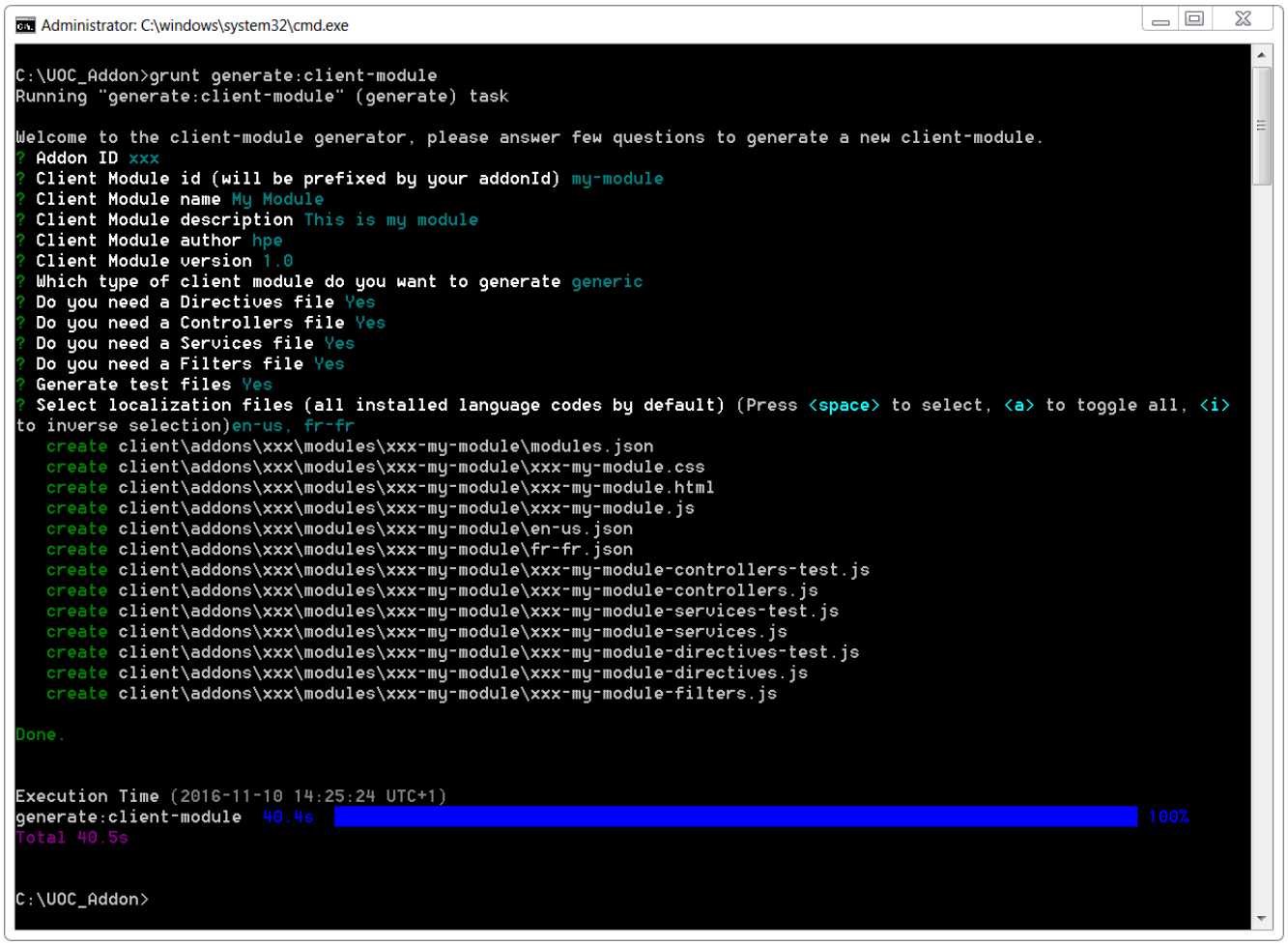


Figure 37: Generator – Generate your generic module

When UOC will restart, the new module component is available and can be browsed through the GUI in the Add-ons – Modules management screen.

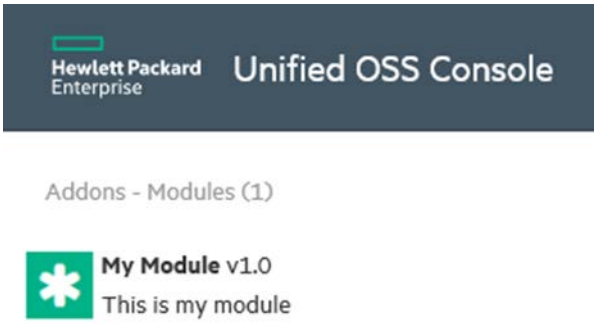


Figure 38: Generator – Add-ons – Modules Management

5.3.2 How to generate an add-on Module Filter

You can use the following command to generate a module component.

```
grunt generate
```

Then select in the menu selection: HPE Add-on client-module -or- enter directly the following command.

```
grunt generate:client-module
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the module identifier. This id will be prefixed by the add-on id in the generated file (separated by a "-")
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the module description
4. Enter the module author or validate the default one you previously entered in your profile
5. Enter the module version or validate the default one you previously entered in your profile
6. Specify '**filter**' in the type of module you want to generate. For this type, you need to cover multiples need, so the generator asks you all the files it can generate for your module.

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Module will be placed under <install_dir>\client\addons\<my add-on Id>\modules\<my add-on id>-<my module id> directory and will be composed of the following files:

Files	Description
modules.json	Descriptor of the module
<my add-on id>-<my module id>.js	Implementation of the main module for the module
<my add-on id>-<my module id>-filters.js	Implementation of filters for the module

NOTE: You can customize a specific module icon in:
<install_dir>\client\public\images\modules\<my add-on id>-<my-module id>.png

Default icon is under <install_dir>\client\public\images\modules\default.png

It is also recommended to review and adjust the descriptor.

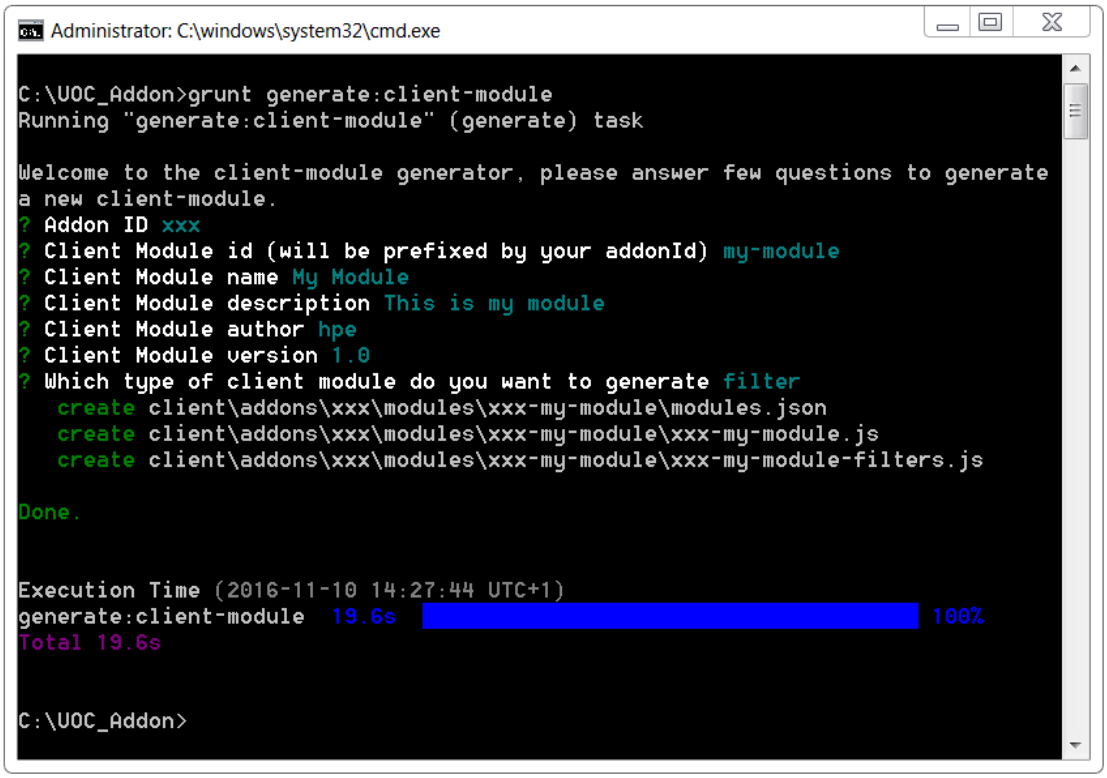


Figure 39: Generator – Generate your filter module

When UOC will restart, the new module component is available and can be browsed through the GUI in the Add-ons – Modules management screen.

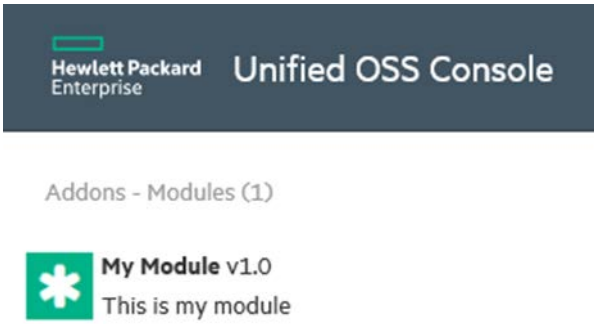


Figure 40: Generator – Add-ons – Modules Management

5.3.3 How to generate an add-on Module Action

You can use the following command to generate a module component.

```
grunt generate
```

Then select in the menu selection: HPE Add-on client-module -or- enter directly the following command.

```
grunt generate:client-module
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the module identifier. This id will be prefixed by the add-on id in the generated file (Separated by a "-")
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the module description
4. Enter the module author or validate the default one you previously entered in your profile
5. Enter the module version or validate the default one you previously entered in your profile
6. Specify '**action**' in the type of module you want to generate. For this type, you need to cover multiples need, so the generator asks you all the files it can generate for your module.
7. Select if you want to generate skeleton files for automatic tests

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Module will be placed under <install_dir>\client\addons\<my add-on Id>\modules\<my add-on id>-<my module id> directory and will be composed of the following files:

Files	Description
modules.json	Descriptor of the module
<my add-on id>-<my module id>.js	Implementation of the main module for the module
<my add-on id>-<my module id>-services.js	Implementation of services for the module

NOTE: You can customize a specific module icon in:
<install_dir>\client\public\images\modules\<my add-on id>-<my-module id>.png

Default icon is under <install_dir>\client\public\images\modules\default.png

It is also recommended to review and adjust the descriptor.

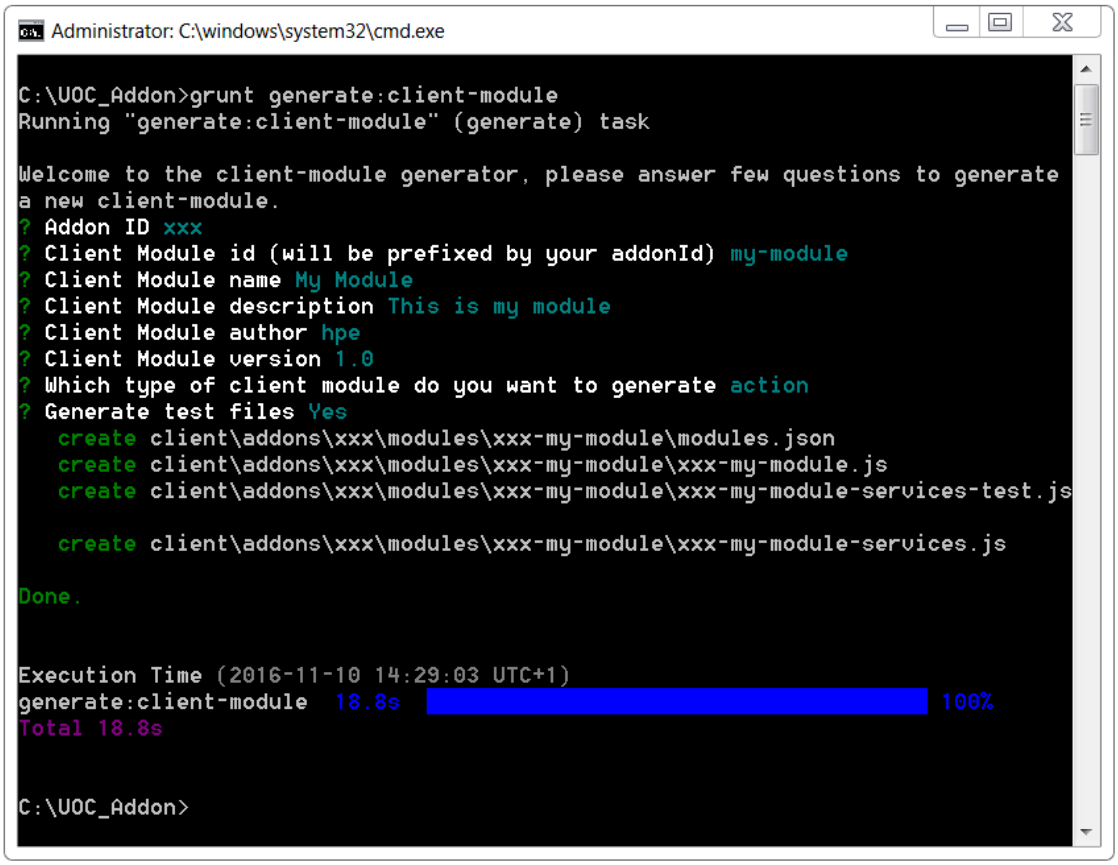


Figure 41: Generator – Generate your action module

When UOC will restart, the new module component is available and can be browsed through the GUI in the Add-ons – Modules management screen.

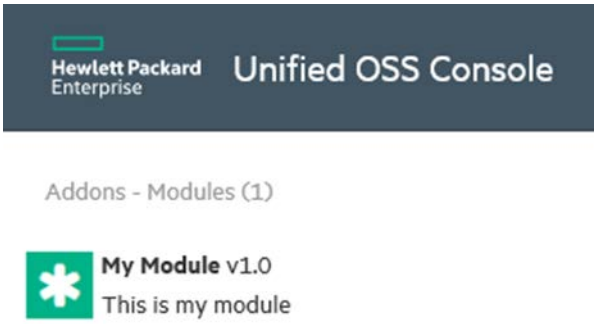


Figure 42: Generator – Add-ons – Modules Management

5.4 How to generate an add-on client Launch Keyword

You can use the following command to generate a launch keyword component.

```
grunt generate
```

Then select in the menu selection: HPE Add-on launch-keyword -or- enter directly the following command.

```
grunt generate:launch-keyword
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the launch keyword identifier. This id will be prefixed by the add-on id in the generated file (Separated by a "-")
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the launch-keyword description
4. Enter the launch-keyword author or validate the default one you previously entered in your profile
5. Enter the launch-keyword version or validate the default one you previously entered in your profile

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Launch keyword will be placed under <install_dir>\client\addons\<my add-on Id>\launch-keywords\<my add-on id>-<my launch-keyword id> directory and will be composed of the following files:

Files	Description
launch-keywords.json	Descriptor of the launch keyword
<my add-on id>-<my launch-keyword id>-services.js	Implementation of the launch keyword as an angular services

NOTE: You can customize a specific launch-keyword icon in: <install_dir>\client\public\images\launch-keywords\<my add-on id>-<my launch-keyword id>.png

Default icon is under <install_dir>\client\public\images\launch-keywords\default.png

It is also recommended to review and adjust the descriptor.

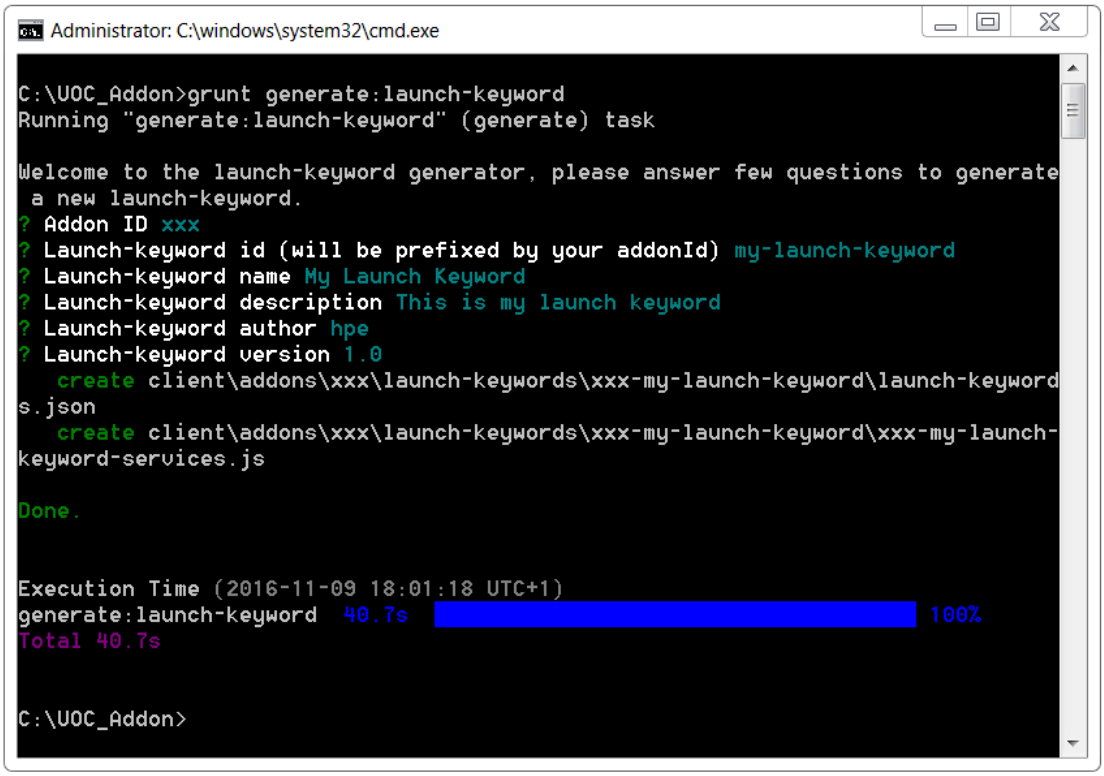


Figure 43: Generator – Generate your launch-keyword

When UOC will restart, the new launch-keyword component is available and can be browsed through the GUI in the Add-ons – Launch-Keywords management screen.

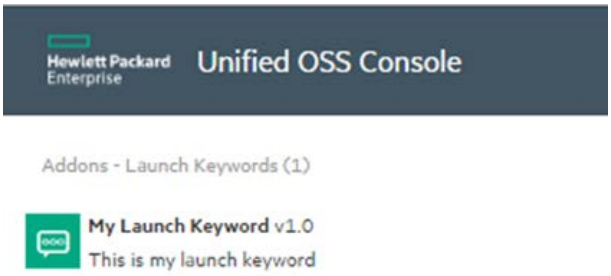


Figure 44: Generator – Add-ons – Launch Keywords Management

5.5 How to generate an add-on client Layout

You can use the following command to generate a layout component.

```
grunt generate
```

Then select in the menu selection: HPE Add-on layout -or- enter directly the following command.

```
grunt generate:layout
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the layout identifier. This id will be prefixed by the add-on id in the generated file (Separated by "-")
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the layout description
4. Enter the layout author or validate the default one you previously entered in your profile
5. Enter the layout version or validate the default one you previously entered in your profile
6. Enter a category to ease organization in View Designer (usually associated to your Add-on Id)
7. Layout wizard steps - Enter the number of widgets you want to place in each row. Press Enter to stop the wizard and start generation.

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Layout will be placed under <install_dir>\client\addons\<my add-on Id>\layouts\<my add-on id>-<my layout id> directory and will be composed of the following files:

Files	Description
layouts.json	Descriptor of the layout
<my add-on id>-<my layout id>.html	Implementation of the layout as HTML file based on bootstrap.

NOTE: You can customize a specific layout icon in:
<install_dir>\client\public\images\layouts\<my add-on id>-<my layout id>.png

Default icon is under <install_dir>\client\public\images\layouts\default.png

It is also recommended to review and adjust the descriptor.

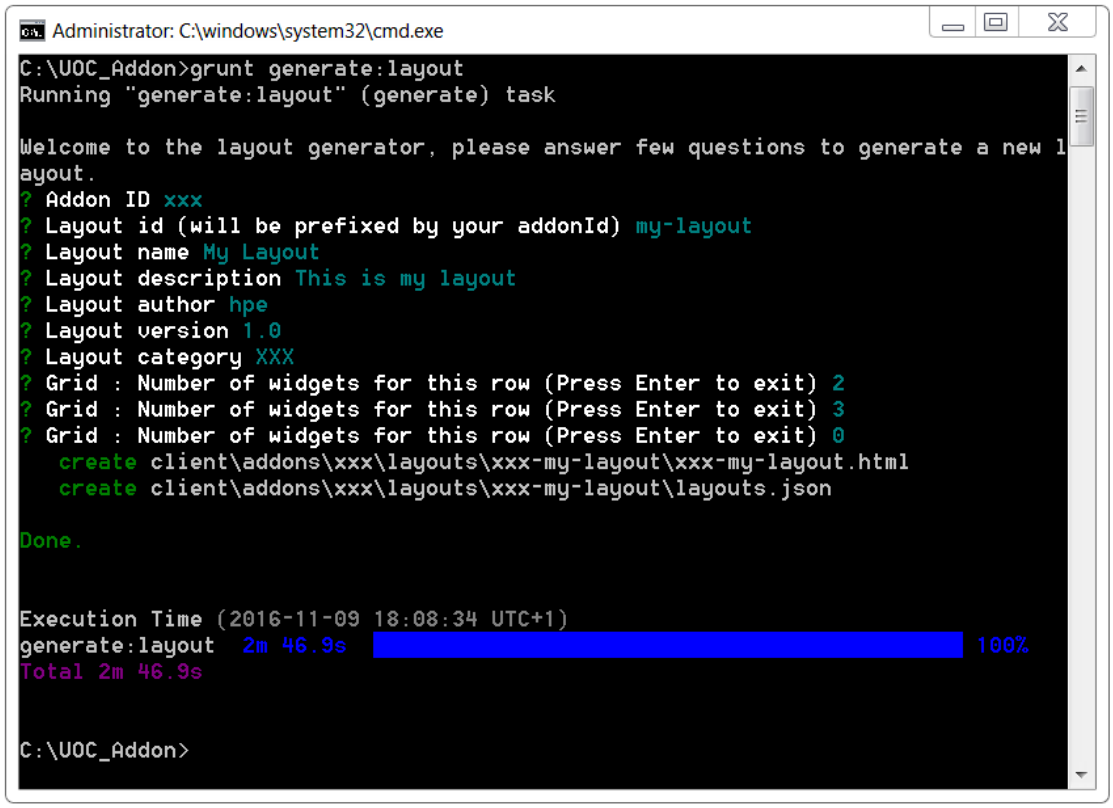


Figure 45: Generator – Generate your layout

When UOC will restart, the new layout component is available and can be browsed through the GUI in the Add-ons – Layouts management screen.

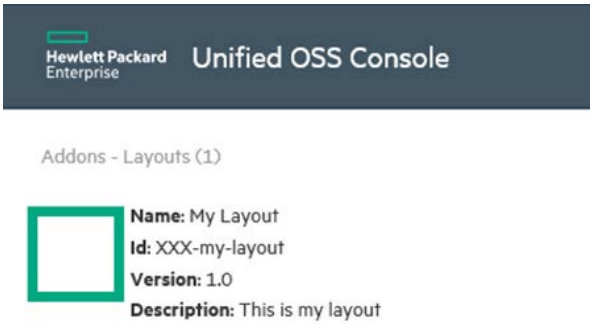


Figure 46: Generator – Add-ons – Layouts Management

5.6 How to generate an add-on client Menu Bar

You can use the following command to generate a layout component.

```
grunt generate
```

Then select in the menu selection: HPE Add-on menu-bar -or- enter directly the following command.

```
grunt generate:menu-bar
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the menu-bar identifier. This id will be prefixed by the add-on id in the generated file (Separated by "-")
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the menu-bar description
4. Enter the menu-bar author or validate the default one you previously entered in your profile
5. Enter the menu-bar version or validate the default one you previously entered in your profile

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Menu-bar will be placed under <install_dir>\client\addons\<my add-on Id>\ menu-bars\<my add-on id>-<my menu-bar id> directory and will be composed of the following files:

Files	Description
menu-bars.json	Descriptor of the menu-bar
<my add-on id>-<my menu-bar id>.html	Implementation of the menu-bar as HTML file based on bootstrap.
<my add-on id>-<my menu-bar id>.css	Implementation of the menu-bar style as CSS.

NOTE: You can customize a specific menu-bar icon in:
<install_dir>\client\public\images\menu-bars\<my add-on id>-<my menu-bar id>.png

Default icon is under <install_dir>\client\public\images\menu-bars\default.png

It is also recommended to review and adjust the descriptor.

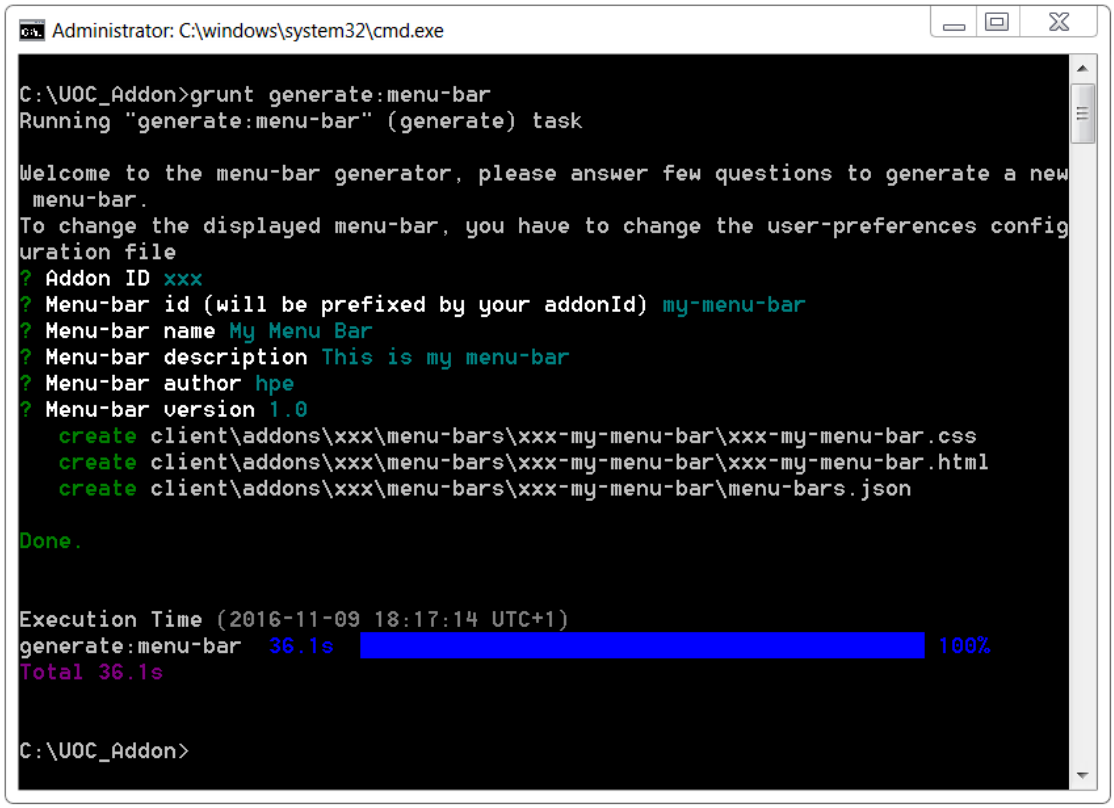


Figure 47: Generator – Generate your menu-bar

When UOC will restart, the new menu-bar component is available and can be browsed through the GUI in the Add-ons – Menu-bars management screen.

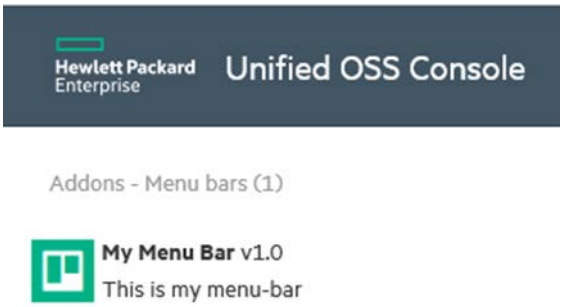


Figure 48: Generator – Add-ons – Menu bars Management

5.7 How to generate an add-on client Menu Item

You can use the following command to generate a menu-item component.

```
grunt generate
```

Then select in the menu selection: HPE Add-on menu-item -or- enter directly the following command.

```
grunt generate:menu-item
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the menu-item identifier. This id will be prefixed by the add-on id in the generated file (Separated by "-")
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the menu- item description
4. Enter the menu- item author or validate the default one you previously entered in your profile
5. Enter the menu- item version or validate the default one you previously entered in your profile
6. Select localization files to generate (all installed language codes by default).

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Menu-item will be placed under <install_dir>\client\addons\<my add-on Id>\ menu- items\<my add-on id>-<my menu-item id> directory and will be composed of the following files:

Files	Description
menu- items.json	Descriptor of the menu- item
<my add-on id>-<my menu- item id>.tpl.html	Definition of the template HTML to describe the HTML new tag to use.
<my add-on id>-<my menu- item id>.html	Implementation of the menu- item as HTML file based on bootstrap.
<my add-on id>-<my menu- item id>.css	Implementation of the menu- item style as CSS.
<my add-on id>-<my menu- item id>.js	Implementation of the main module for the menu item
<my add-on id>-<my menu- item id>-controllers.js	Implementation of controllers for the menu item
<my add-on id>-<my menu- item id>-directives..js	Implementation of directives for the menu item
<language code>.json	One localization JSON file per supported language codes. *(e.g. en-us.json, fr-fr.json,...)*

NOTE: You can customize a specific menu- item icon in:
<install_dir>\client\public\images\menu-items\<my add-on id>-<my menu- item id>.png

Default icon is under <install_dir>\client\public\images\menu-items\default.png

It is also recommended to review and adjust the descriptor.

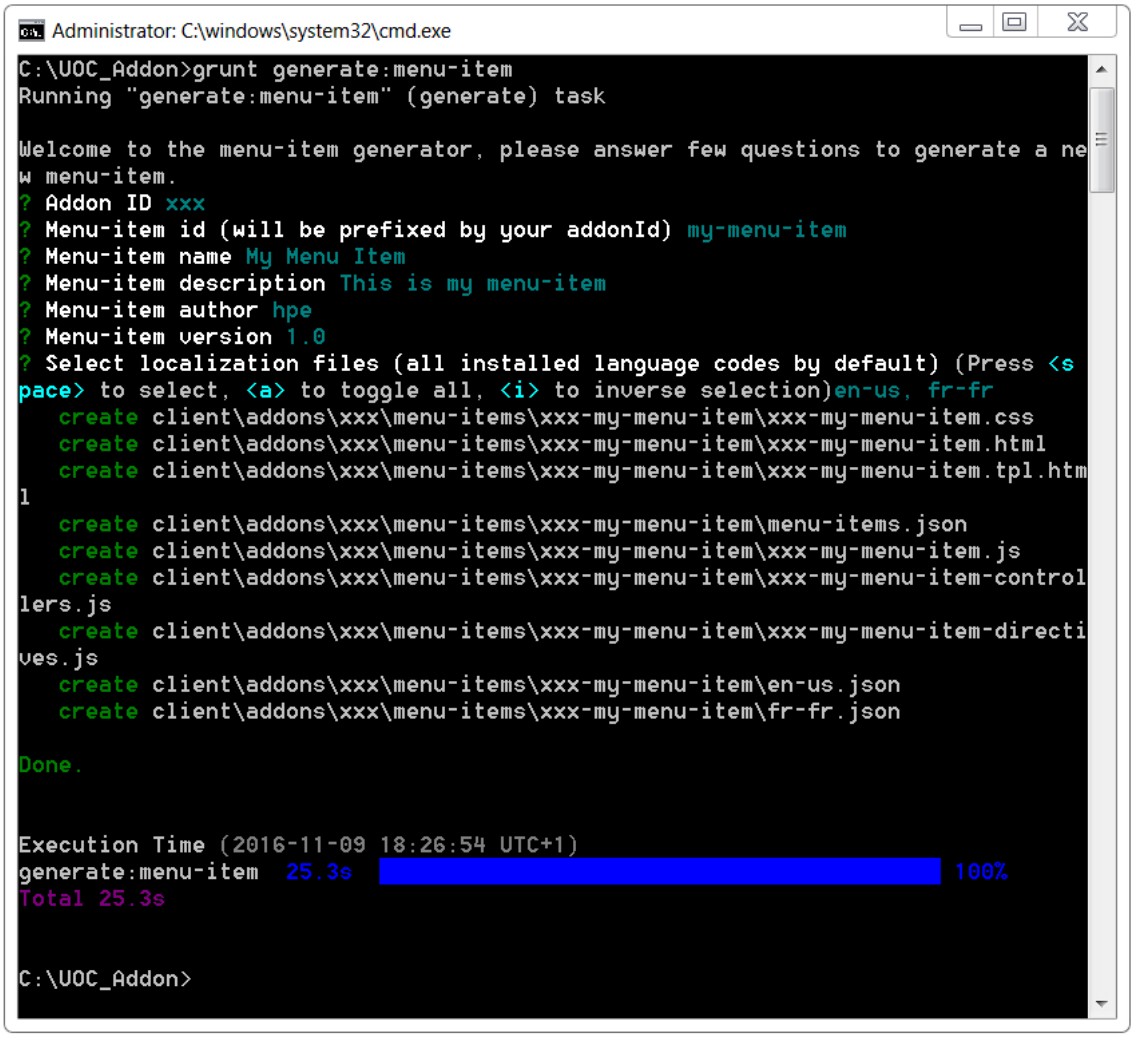


Figure 49: Generator – Generate your menu- item

When UOC will restart, the new menu- item component is available and can be browsed through the GUI in the Add-ons – Menu- items management screen.

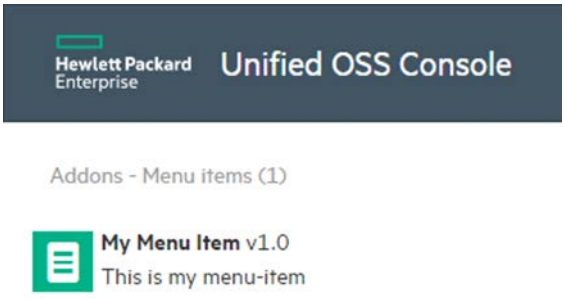


Figure 50: Generator – Add-ons – Menu items Management

5.8 How to generate an add-on server Package

You can use the following command to generate a package associated to a plugin.

```
grunt generate
```

Then select in the menu selection: HPE Add-on package -or- enter directly the following command.

```
grunt generate:package
```

1. Enter the Plugin Identifier you want to associate this new package to. (usually it is the same as your registered Add-on ID)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the package identifier
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the package description
4. Enter the package author or validate the default one you previously entered in your profile
5. Enter the package version or validate the default one you previously entered in your profile

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Package will be placed under its associated plugin <install_dir>\server\public\addons\<my plugin Id>\packages\<my package id> directory and will be composed of the following files:

Files	Description
package.json	Meta-data of the package that defines all the information this package can manage. *(dimensions, facts, objets, relations, etc.)*
gui	GUI directory of the package
• views	Directory to store all the views
• view.json	Definition of the graphical views available for this package
• workspaces	Directory to store all the workspaces
• workspace.json	Definition of the graphical workspaces available for this package

NOTE: You can customize a specific package icon in:
<install_dir>\client\public\images\packages\<my package id>.png

Default icon is under <install_dir>\client\public\images\packages\default.png

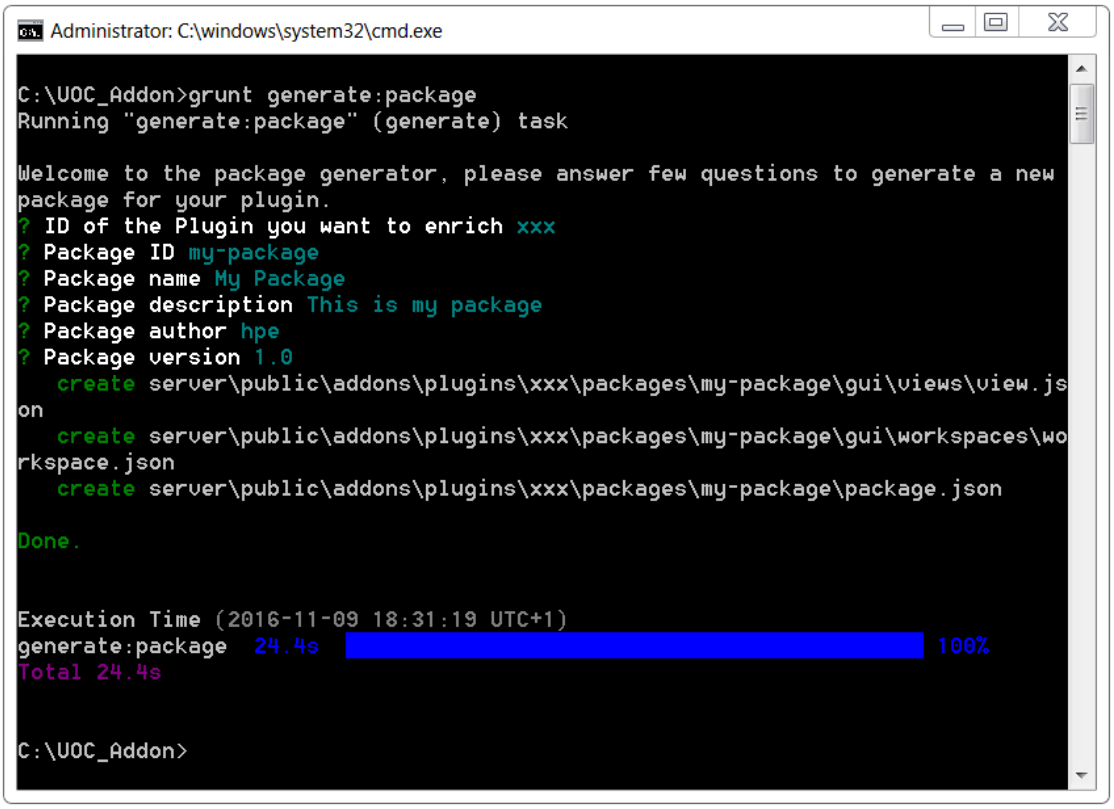


Figure 51: Generator – Generate your package for a plugin

When UOC will restart, the new package is available and can be browsed through the GUI in the Add-ons – Plugins, then click on a plugin to check its packages. You also can browse directly all the packages group by plugin in the menu Packages

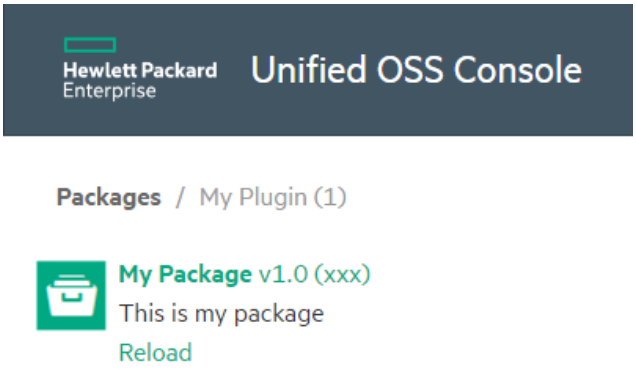


Figure 52: Generator – Add-ons – Plugins Management

5.9 How to generate an add-on server Plugin

You can use the following command to generate a plugin to implement the connection between uoc and your data server(s).

```
grunt generate
```

Then select in the menu selection: **HPE Add-on plugin** -or- enter directly the following command.

```
grunt generate:plugin
```

1. Enter the plugin identifier (Usually it equals your registered Add-on ID)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the plugin name
3. Enter the plugin description
4. Enter the plugin author or validate the default one you previously entered in your profile
5. Enter the plugin version or validate the default one you previously entered in your profile
6. Ask if you also want to generate a new package at the same time associated to this plugin (Y/N). If yes, please refer to the section 5.8 How to generate an add-on server Package for more details.

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Plugin is composed of two parts:

- Implementation of the plugin that will be place under <install_dir>\server\addons\<my plugin Id> directory and will be composed of the following files:

Files	Description
plugin.json	Descriptor of the plugin
constants.js	Definition of the constant used in this plugin
routes.js	Definition of all URL supported by this plugin. These URL must follow the unified format requested by a plugin.
package-manager.js	Implementation of the routes (URL) to provide request yoru data server, format response and send unified information to uoc.
package.json	NPM's package definiton to define external dependencies and versions
README.txt	Simple read me file to add details about your plugin.

- Configuration file of the plugin that will be place under <install_dir>\server\public\addons\<my plugin Id> directory and will be composed of the following files:

Files	Description
config.json	Configuration file used during development.
config.json.kitting	Configuration file used for packaging and kitting

IMPORTANT: The code of the plugin can be replaced by a patch or upgrade, but files store in <install_dir>/data, <install_dir>/client/public or <install_dir>/server/public are never uninstall or replaced.

This grant that after an upgrade of the product, there is not lost of configuration, or customization.

TIP: A file with the extension .kitting is dedicated to be used during the packaging and kitting process to repace a file used only during development.

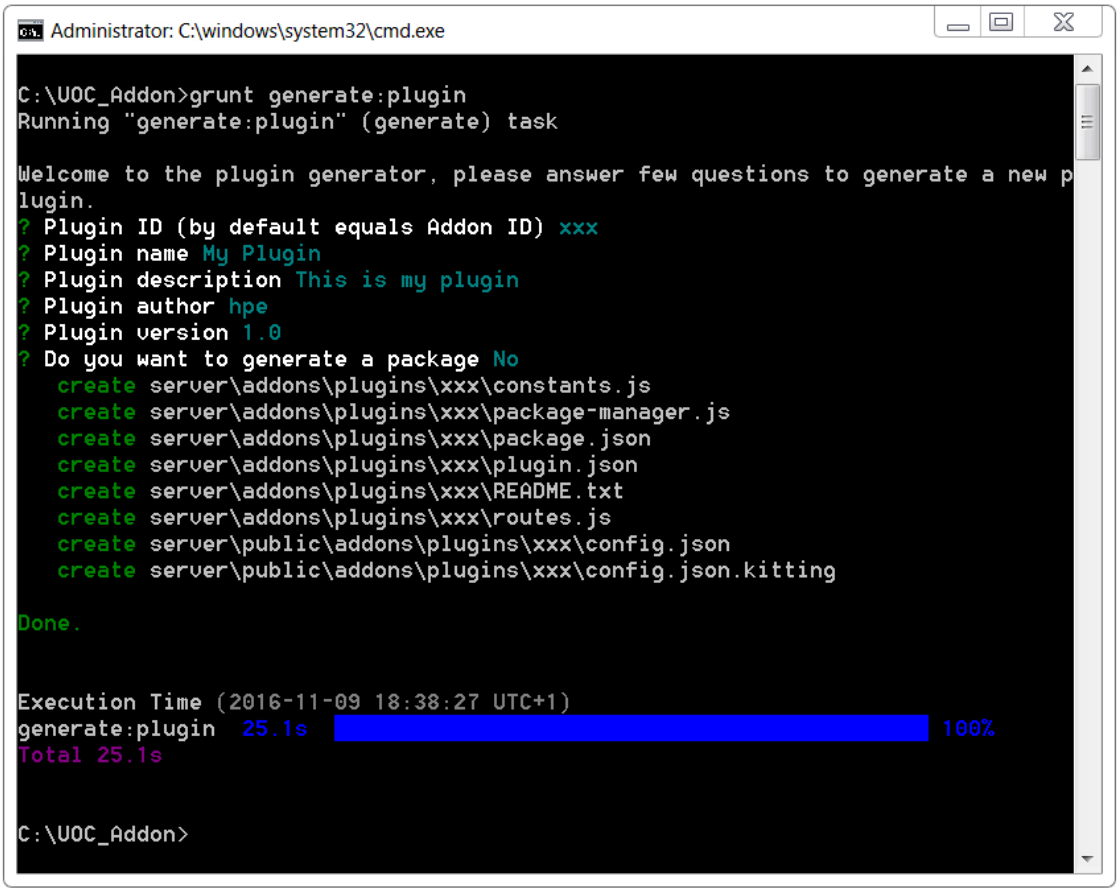


Figure 53: Generator – Generate your package for a plugin

When UOC will restart, the new plugin is available and can be browsed through the GUI in the Add-ons – Plugins.

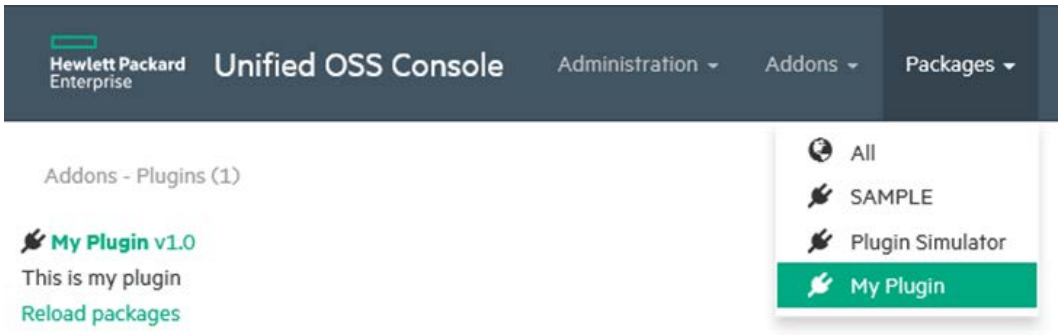


Figure 54: Generator – Add-ons – Plugins Management

5.10 How to generate an add-on client Theme

You can use the following command to generate a theme component.

```
grunt generate
```

Then select in the menu selection: HPE Add-on - theme -or- enter directly the following command.

```
grunt generate:theme
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the theme identifier. This id will be prefixed by the add-on id in the generated file (Separated by "-")
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the theme description
4. Enter the theme author or validate the default one you previously entered in your profile
5. Enter the theme version or validate the default one you previously entered in your profile
6. Enter a category to ease organization in View Designer (usually associated to your Add-on Id)
7. Theme wizard steps – Select the existing theme to use as a skeleton for this new theme (hpe dark or light theme)
8. Theme wizard steps – Customize the Bootstrap variables. Answer No to skip this step, else enter color to setup for several variable.

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Theme will be placed under <install_dir>\client\addons\<my add-on Id>\themes\<my add-on id>-<my theme id> directory and will be composed of the following files:

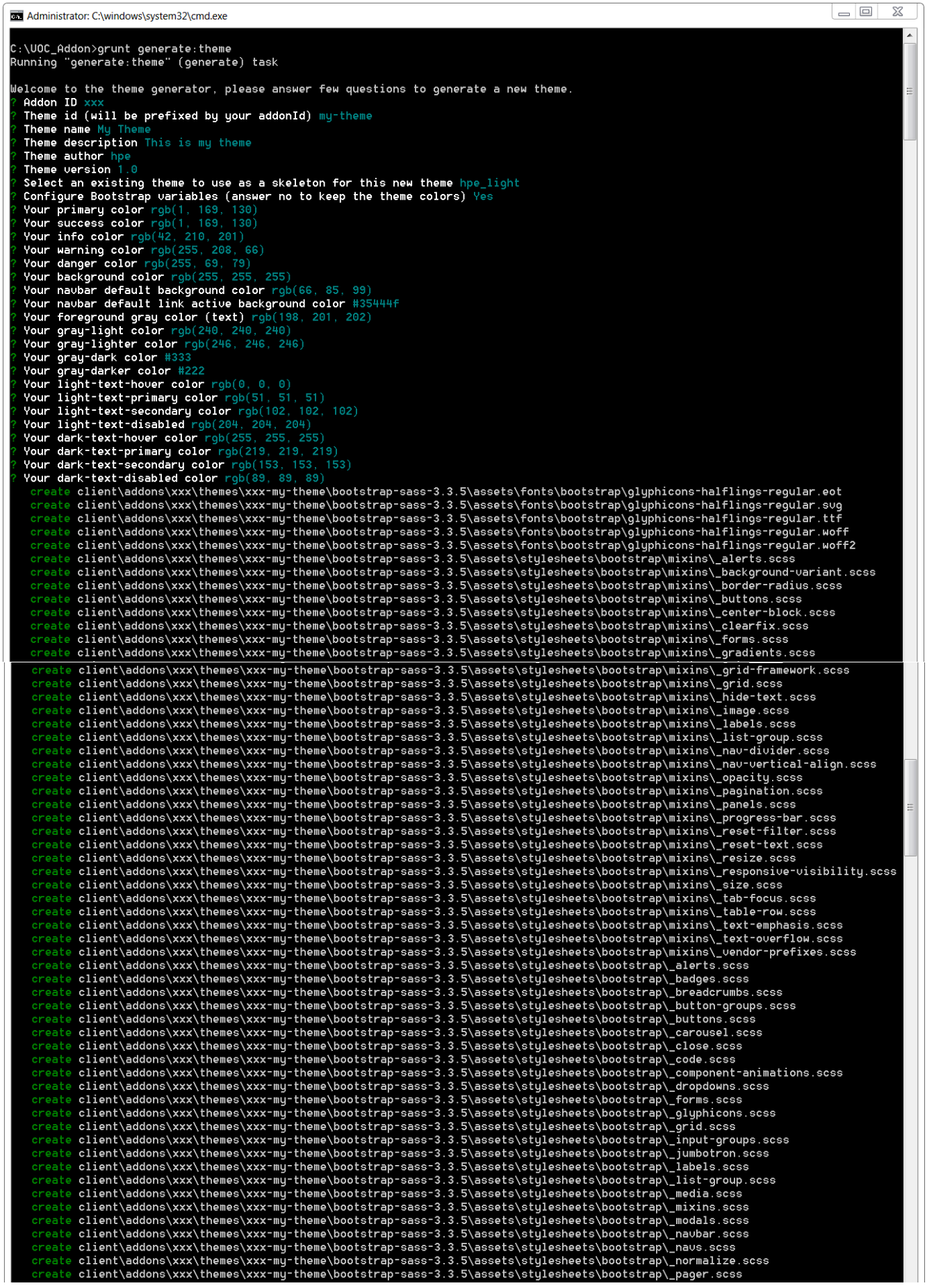
Files	Description
themes.json	Descriptor of the theme
compile-sass.bat	Script for Windows to compile Sass files of the theme and generate CSS files
compile-sass.sh	Script for Linux to compile Sass files of the theme and generate CSS files
css	After compilation of the theme, this directory is generated with CSS files used at runtime.
scss	Sass CSS directory
• components	Directory where each UOC component can optionally define a specific customization to override the style implemented by default. *Naming: <addon-id>-<component id>.scss*
• _hpe-variables.scss	Color variables extracted from the HPE brand. *Note: This file is included in other files and never compiled directly.*
• _bootstrap-variables.scss	Bootstrap variables. Uncomment and replace the mto cusotmize the default bootstrap theme. *Note: This file is included in other files and never compiled directly.*
• <my add-on id>-<my theme id>.scss	Bootstrap customization not only possible using variables. It overrides bootstrap selectors. *Note: This file contains only modifications of Bootstrap.*
• main.scss	UOC specific customizations that are not part of Bootstrap (helper classes, fixes, etc.)
images	Directory to store specific images for the theme

• branding	Directory to store images to customize the branding of the theme.
o favicon.ico	Favicon to display in the web browser associated to the URL of the application (must be 16 x 16)
o <theme id>.png	Icon of the theme to use in add-ons management. (must be 48 x 48)
o login.png	Icon to use for the login page (local authentication only). Identity Provider provides the login page for SAML authentication.
o logo.png	Icon to use as logo in the main menu bar, next to the title. (recommended size: width x 136)
• wallpapers	Directory to store wallpapers of the theme
o wallpaper.png	Active wallpaper for the theme used for the login page (local authentication only)
highcharts	Directory specific for Highcharts product.
• highcharts.json	Specific theme part for Highcharts graphical library. (JSON format) Check these links for more details https://github.com/highcharts/highcharts/tree/master/js/themes http://www.highcharts.com/docs/chart-design-and-style/themes
fonts	Directory to store font families of the theme. By default you will have HPE Simple and MetricHPE (font dedicated to HPE product).
bootstrap-sass-3.3.x	Directory to store official Sass part of bootstrap.

NOTE: You can customize a specific theme icon in:
<install_dir>\client\public\images\themes\<my add-on id>-<my theme id>.png

Default icon is under <install_dir>\client\public\images\themes\default.png

It is also recommended to review and adjust the descriptor.


Administrator: C:\windows\system32\cmd.exe
C:\UOC_Addon>grunt generate:theme
Running "generate:theme" (generate) task
Welcome to the theme generator, please answer few questions to generate a new theme.
? Addon ID xxx
? Theme id (will be prefixed by your addonId) my-theme
? Theme name My Theme
? Theme description This is my theme
? Theme author hpe
? Theme version 1.0
? Select an existing theme to use as a skeleton for this new theme hpe_light
? Configure Bootstrap variables (answer no to keep the theme colors) Yes
? Your primary color rgb(1, 169, 130)
? Your success color rgb(1, 169, 130)
? Your info color rgb(42, 210, 201)
? Your warning color rgb(255, 208, 66)
? Your danger color rgb(255, 69, 79)
? Your background color rgb(255, 255, 255)
? Your navbar default background color rgb(66, 85, 99)
? Your navbar default link active background color #35444f
? Your foreground gray color (text) rgb(198, 201, 202)
? Your gray-light color rgb(240, 240, 240)
? Your gray-lighter color rgb(246, 246, 246)
? Your gray-dark color #333
? Your gray-darker color #222
? Your light-text-hover color rgb(0, 0, 0)
? Your light-text-primary color rgb(51, 51, 51)
? Your light-text-secondary color rgb(102, 102, 102)
? Your light-text-disabled rgb(204, 204, 204)
? Your dark-text-hover color rgb(255, 255, 255)
? Your dark-text-primary color rgb(219, 219, 219)
? Your dark-text-secondary color rgb(153, 153, 153)
? Your dark-text-disabled color rgb(89, 89, 89)
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\fonts\bootstrap\glyphicons-halflings-regular.eot
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\fonts\bootstrap\glyphicons-halflings-regular.svg
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\fonts\bootstrap\glyphicons-halflings-regular.ttf
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\fonts\bootstrap\glyphicons-halflings-regular.woff
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\fonts\bootstrap\glyphicons-halflings-regular.woff2
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_alerts.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_background-variant.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_border-radius.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_buttons.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_center-block.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_clearfix.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_forms.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_gradients.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_grid-framework.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_grid.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_hide-text.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_image.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_labels.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_list-group.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_nav-divider.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_nav-vertical-align.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_opacity.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_pagination.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_panels.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_progress-bar.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_reset-filter.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_reset-text.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_resize.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_responsive-visibility.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_size.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_tab-focus.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_table-row.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_text-emphasis.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_text-overflow.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap\mixins_vendor-prefixes.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_alerts.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_badges.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_breadcrumbs.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_button-groups.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_buttons.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_carousel.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_close.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_code.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_component-animations.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_dropdowns.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_forms.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_glyphicons.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_grid.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_input-groups.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_jumbotron.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_labels.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_list-group.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_media.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_mixins.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_modals.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_navbar.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_navs.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_normalize.scss
create client\addons\xxx\themes\xxx-my-theme\bootstrap-sass-3.3.5\assets\stylesheets\bootstrap_pager.scss

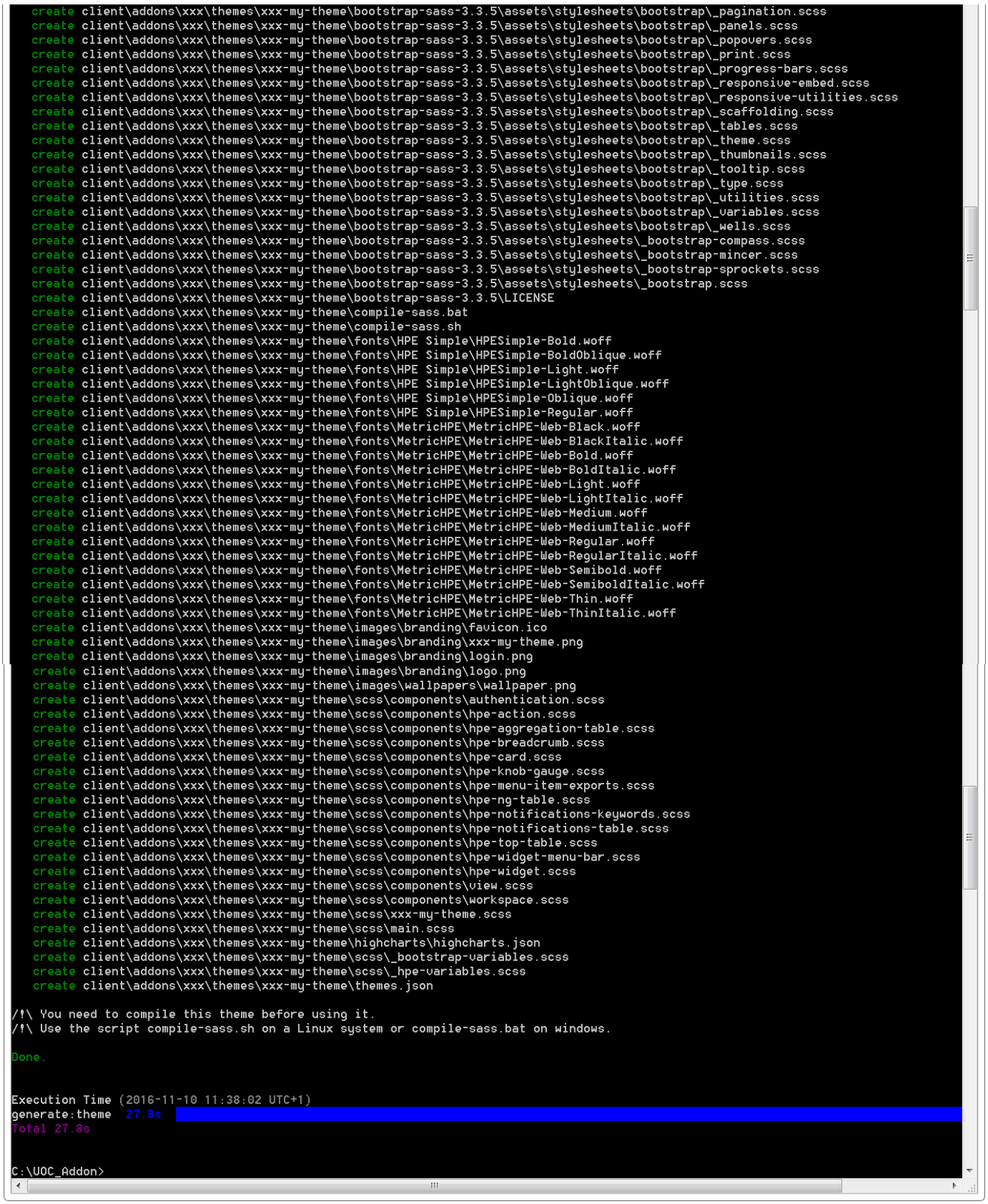


Figure 55: Generator - Generate your theme

You will need to compile the Sass CSS files of the theme to generate the CSS used at runtime. In order to do this, you need to browse your generated theme directory <install_dir>\client\addons\<my add-on Id>\themes\<my add-on id>-<my theme id> and run the following script.

On Windows:

```
compile-sass.bat
```

On Linux:

```
$ compile-sass.sh
```

When UOC will restart, the new theme component is available and can be browsed through the GUI in the Add-ons – Themes management screen and it is also listed in your user preferences menu.

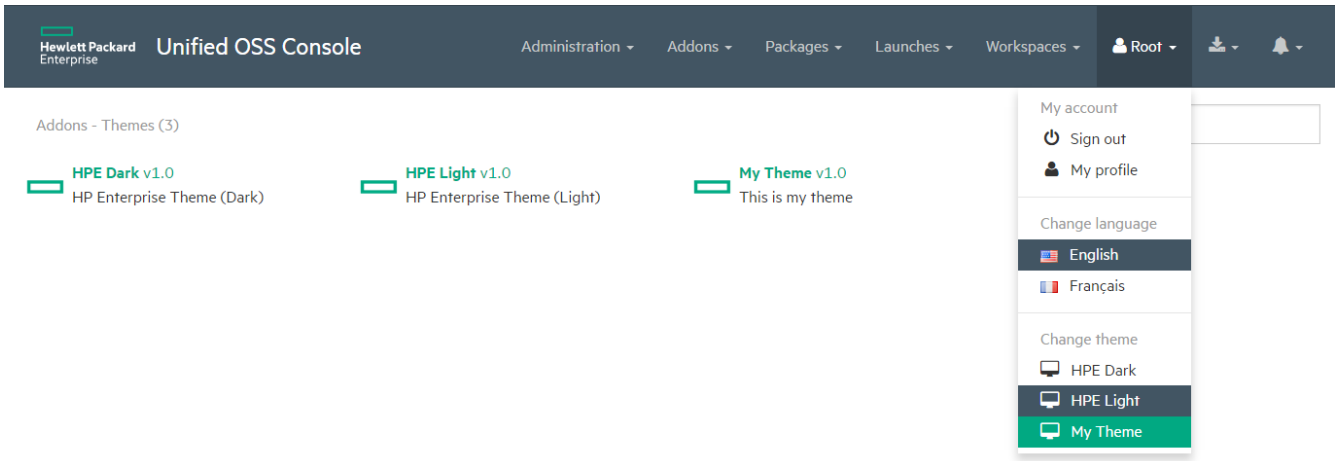


Figure 56: Generator – Add-ons – Themes Management

You can try it by clicking on the theme icon or by selecting its name in the user preferences menu.

5.11 How to generate an add-on client Widget

You can use the following command to generate a widget component.

```
grunt generate
```

Then select in the menu selection: HPE Add-on widget -or- enter directly the following command.

```
grunt generate:widget
```

1. Enter the Add-on Identifier i.e. the registered Id you submit to the support team (see 4.2 Add-on ID Registration)
   Format: Only letters and "-" are allowed. It must start and end with a letter
2. Enter the widget identifier. This id will be prefixed by the add-on id in the generated file. (Separated by "-")
   Format: Only letters and "-" are allowed. It must start and end with a letter
3. Enter the widget description
4. Enter the widget author or validate the default one you previously entered in your profile
5. Enter the widget version or validate the default one you previously entered in your profile
6. Enter a category to ease organization in View Designer (usually associated to your Add-on Id)
7. Select if you want to generate skeleton files for automatic tests
8. Select localization files to generate (all installed language codes by default).

TIP: Use arrow keys to navigate, you can use <space> to select/unselect, <i> to toggle the selection, <a> to select all.

Widget will be placed under <install_dir>\client\addons\<my add-on Id>\widgets\<my add-on id>-<my widget id> directory and will be composed of the following files:

Files	Description
widgets.json	Descriptor of the widget
<my add-on id>-<my widget id>.json	JSON Schema to configure this widget
<my add-on id>-<my widget id>.tpl.html	Definition of the template HTML to describe the HTML new tag to use.
<my add-on id>-<my widget id>.html	Implementation of the widget as HTML file based on bootstrap.
<my add-on id>-<my widget id>init.html	Implementation of the widget as HTML file based on bootstrap.
<my add-on id>-<my widget id>.cfg.html	Implementation of the widget configuration popup content as HTML file based on bootstrap.
<my add-on id>-<my widget id>.css	Implementation of the widget style as CSS.
<my add-on id>-<my widget id>.js	Implementation of the main module for the widget
<my add-on id>-<my widget id>-controllers.js	Implementation of controllers for the widget
<my add-on id>-<my widget id>-controllers-test.js	Controllers unit test file validating that controllers can be loaded without error. (if tests were generated)
<my add-on id>-<my widget id>-directives.js	Implementation of directives for the widget
<my add-on id>-<my widget id>-directives-test.js	Directives unit test file validating that directives can be loaded without error. (if tests were generated)
<my add-on id>-<my widget id>-services.js	Implementation of services for the widget
<my add-on id>-<my widget id>-services-test.js	Services unit test file validating that services can be loaded without error. (if tests were generated)
<language code>.json	One localization JSON file per supported language codes. *(e.g. en-us.json, fr-fr.json,...)*

NOTE: You can customize a specific widget icon in:
<install_dir>\client\public\images\widgets\<my add-on id>-<my-widget id>.png

Default icon is under <install_dir>\client\public\images\widgets\default.png

It is also recommended to review and adjust the descriptor.

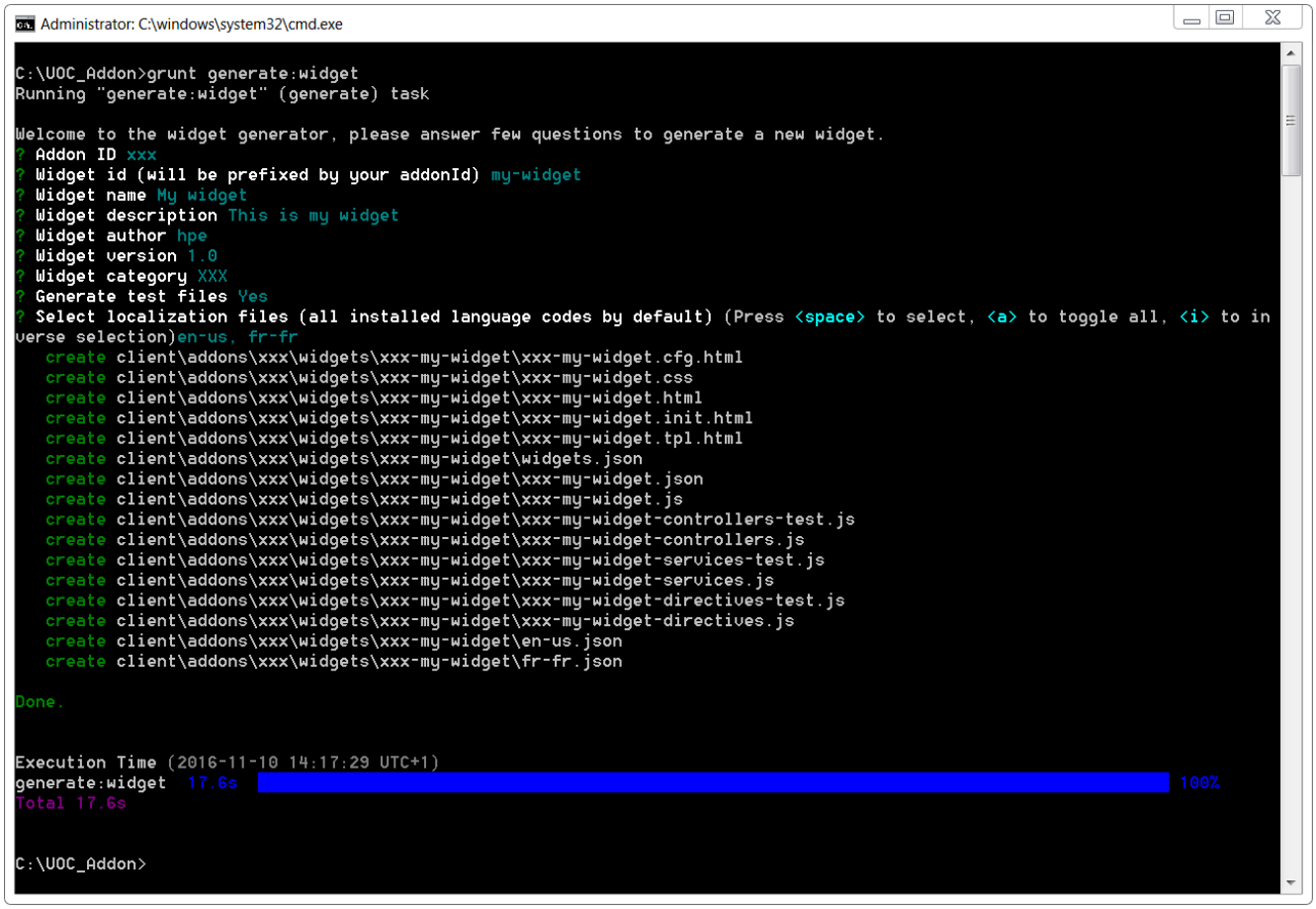


Figure 57: Generator – Generate your menu- item

When UOC will restart, the new widget component is available and can be browsed through the GUI in the Add-ons – Widgets management screen.

Figure 58: Generator – Add-ons – Widget Management

Chapter 6
UOC JSON Schema

JSON Schema is a vocabulary that allows you describe the existing data format and ease the validation of JSON documents. The latest IETF published draft is v4.

NOTE: Please refer to JSON schema specification and JSON schema (http://json-schema.org)

All these JSON Schema are available in the following location: **<install_dir>/json-schema**

6.1 Packages

The object user has the following definition in JSON. **<install_dir>/json-schema/packages.json**

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "array",
  "title": "Packages",
  "description": "The packages.json is a file where one can describe the
UI metadata of packages",
  "items": {
    "type": "object",
    "title": "Package definition",
    "description": "An item of this array is a description of UI
metadata of one package",
    "properties": {
      "id": {
        "type": "string",
        "title": "Id",
        "description": "Unique identifier for the package"
      },
      "name": {
        "type": "string",
        "title": "Name",
        "description": "Presentation name of the package"
      },
      "description": {
        "type": "string",
        "title": "Description",
        "description": "Description of the package"
      },
      "version": {
        "type": "string",
```

```
        "title": "Version",
        "description": "Version of the package"
      },
      "author": {
        "type": "string",
        "title": "Author",
        "description": "Author of the package"
      },
      "domain": {
        "type": "string",
        "title": "Domain",
        "description": "Domain of the package"
      },
      "active": {
        "type": "boolean",
        "title": "Active",
        "description": "When set to true, the package will be available
on the client side"
      },
      "roles": {
        "type": "array",
        "title": "Roles",
        "description": "Set of role ids. It is used to restrict the
visibility and the use of this package. When 'roles' is empty, there are
no restriction over the package",
        "items": {
          "type": "string",
          "title": "Role ID",
          "description": "Role ID"
        }
      },
      "objectTypes": {
        "type": "array",
        "title": "ObjectTypes",
        "description": "This set describes the objectTypes metadata",
        "items": {
          "type": "object",
          "title": "ObjectType",
          "description": "Each item describes an objectType metadata",
          "properties": {
            "id": {
              "type": "string",
```

```
              "title": "Id",
              "description": "The unique identifier of the objectType"
            },
            "name": {
              "type": "string",
              "title": "Name",
              "description": "The name of the objectType"
            },
            "version": {
              "type": "string",
              "title": "Version",
              "description": "The version of the objectType"
            },
            "description": {
              "type": "string",
              "title": "Description",
              "description": "Description of the objectType"
            },
            "key": {
              "type": "array",
              "title": "Key",
              "description": "This key is a set of property id on which
we certify the unicity of an object instance",
              "items": {
                "type": "string",
                "title": "Property",
                "description": "Property of the object that will be
checked for unicity"
              }
            },
            "roles": {
              "type": "array",
              "title": "Roles",
              "description": "Set of role ids. It is used to restrict
the visibility and the use of this objectType. When 'roles' is empty,
there are no restriction over the objectType",
              "items": {
                "type": "string",
                "title": "Role ID",
                "description": "Role ID"
              }
            },
            "operations": {
```

```
              "type": "array",
              "title": "Operations",
              "description": "Operations related to this objectType",
              "items": {
                "type": "object",
                "title": "Operation",
                "description": "Each item describes an operation",
                "properties": {
                  "id": {
                    "type": "string",
                    "title": "Id",
                    "description": "Unique identifier for the operation"
                  },
                  "name": {
                    "type": "string",
                    "title": "Name",
                    "description": "Name of the operation"
                  },
                  "roles": {
                    "type": "array",
                    "title": "Roles",
                    "description": "Set of role ids. It is used to
restrict the visibility of this operation. When 'roles' is empty, there
are no restriction on the operation",
                    "items": {
                      "type": "string",
                      "title": "Role ID",
                      "description": "Role ID"
                    }
                  }
                },
                "required": [
                  "id",
                  "name"
                ]
              }
            }
          },
          "required": [
            "id",
            "name",
            "version",
```

```
            "description",
            "key",
            "operations"
          ]
        }
      },
      "dimensionTree": {
        "type": "object",
        "title": "DimensionTree",
        "description": "This object describes the dimensions' metadata",
        "properties": {
          "folders": {
            "type": "array",
            "title": "Folders",
            "description": "Describes the folder tree where the
dimensions are located",
            "items": {
              "type": "object",
              "title": "Folder",
              "description": "A Folder in which you can find description
of dimensions and/or sub folders",
              "properties": {
                "name": {
                  "type": "string",
                  "title": "Name",
                  "description": "Presentation name of the folder"
                },
                "description": {
                  "type": "string",
                  "title": "Description",
                  "description": "Description of the folder"
                },
                "folders": {
                  "$ref": "#folders"
                },
                "dimensions": {
                  "type": "array",
                  "title": "Dimensions",
                  "description": "Definition of the dimensions located
in this folder",
                  "items": {
                    "type": "object",
```

```
                    "title": "Dimension definition",
                    "description": "Definition of the dimension",
                    "properties": {
                      "id": {
                        "type": "string",
                        "title": "Id",
                        "description": "Unique identifier of the
dimension"
                      },
                      "name": {
                        "type": "string",
                        "title": "Name",
                        "description": "Presentation name of the
dimension"
                      },
                      "description": {
                        "type": "string",
                        "title": "Description",
                        "description": "Description of the dimension"
                      },
                      "type": {
                        "type": "string",
                        "title": "Type",
                        "description": "Type of the dimension",
                        "enum": [
                          "NUMBER",
                          "STRING"
                        ]
                      },
                      "hidden": {
                        "type": "boolean",
                        "title": "Hidden",
                        "description": "If the dimension should be
displayed or not"
                      },
                      "lowCardinality": {
                        "type": "boolean",
                        "title": "LowCardinality",
                        "description": "If the dimension is considered
as a low cardinality one"
                      }
                    },
                    "required": [
```

```
                    "id",
                    "name",
                    "description",
                    "type",
                    "lowCardinality"
                  ]
                }
              }
            },
            "required": [
              "name",
              "description",
              "dimensions"
            ]
          }
        }
      },
      "required": [
        "folders"
      ]
    },
    "factTree": {
      "type": "object",
      "title": "FactTree",
      "description": "This object describes the facts' metadata",
      "properties": {
        "folders": {
          "type": "array",
          "title": "Folders",
          "description": "Describes the folder tree where the facts
are located",
          "items": {
            "type": "object",
            "title": "Folder",
            "description": "A Folder in which you can find description
of dimensions and/or sub folders",
            "properties": {
              "name": {
                "type": "string",
                "title": "Name",
                "description": "Presentation name of the folder"
              },
```

```
                "description": {
                  "type": "string",
                  "title": "Description",
                  "description": "Description of the folder"
                },
                "folders": {
                  "$ref": "#folders"
                },
                "facts": {
                  "type": "array",
                  "title": "Facts",
                  "description": "Definition of the facts located in
this folder",
                  "items": {
                    "type": "object",
                    "title": "Fact definition",
                    "description": "Definition of the fact",
                    "properties": {
                      "id": {
                        "type": "string",
                        "title": "Id",
                        "description": "Unique identifier of the fact"
                      },
                      "name": {
                        "type": "string",
                        "title": "Name",
                        "description": "Presentation name of the fact"
                      },
                      "description": {
                        "type": "string",
                        "title": "Description",
                        "description": "Description of the fact"
                      },
                      "type": {
                        "type": "string",
                        "title": "Type",
                        "description": "Type of the fact",
                        "enum": [
                          "NUMBER",
                          "STRING"
                        ]
                      },
```

```
                    "unit": {
                      "type": "string",
                      "title": "Unit",
                      "description": "Unit of the fact"
                    },
                    "worstOrdering": {
                      "type": "string",
                      "title": "WorstOrdering",
                      "description": "What is the value order of this
fact. ASC means lower values are the best, higher values are the worst.
DESC means higher values are the best, lower values are the worst",
                      "enum": [
                        "ASC",
                        "DESC"
                      ]
                    }
                  },
                  "required": [
                    "id",
                    "name",
                    "description",
                    "type",
                    "unit",
                    "worstOrdering"
                  ]
                }
              }
            },
            "required": [
              "name",
              "description",
              "facts"
            ]
          }
        }
      },
      "required": [
        "folders"
      ]
    },
    "relations": {
      "type": "array",
```

```
        "title": "Relations",
        "description": "Relations between dimensions and fact. One or
more facts can be linked to one or more dimensions. vice versa",
        "items": {
          "type": "object",
          "title": "Relation",
          "description": "A relation describes a set of facts and a set
of dimensions",
          "properties": {
            "dimensions": {
              "type": "array",
              "title": "Dimensions",
              "description": "dimensions' ids for the relations",
              "items": {
                "type": "string",
                "title": "Id of the dimension",
                "description": "This id must be defined in the
dimensionTree above"
              }
            },
            "facts": {
              "type": "array",
              "title": "Facts",
              "description": "facts' ids for the relations",
              "items": {
                "type": "string",
                "title": "Id of the fact",
                "description": "This id must be defined in the factTree
above"
              }
            }
          },
          "required": [
            "dimensions",
            "facts"
          ]
        }
      },
      "notifications": {
        "$ref": "#notifications"
      }
    },
    "required": [
```

```
      "id",
      "name",
      "description",
      "version",
      "author",
      "domain",
      "active"
    ]
  }
}
```

6.2 Users

The object user has the following definition in JSON. <install_dir>/json-schema/users.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "array",
  "title": "Users",
  "description": "List of UOC users (local authentication)",
  "items": {
    "type": "object",
    "title": "User",
    "description": "Definition of a user",
    "properties": {
      "user_id": {
        "type": "string",
        "title": "User Identifier",
        "description": "Unique identifier for the user"
      },
      "name": {
        "type": "string",
        "title": "Name",
        "description": "Presentation name of the user"
      },
      "tenant_id": {
        "type": "string",
        "title": "Tenant Identifier",
        "description": " Unique identifier of the tenant if the user is
associated to a specific tenant"
      },
```

```
      "password": {
        "type": "string",
        "title": "Password",
        "description": "Password of the user"
      },
      "roles": {
        "type": "array",
        "title": "Roles",
        "description": "List of roles associated to the user",
        "items": {
          "type": "string",
          "title": "Role",
          "description": "Role Identifier to associate to the user"
        }
      },
      "description": {
        "type": "string",
        "title": "Description",
        "description": "Description of the user"
      }
    },
    "required": [ "user_id", "name", "password", "roles" ]
  }
}
```

6.3 Roles

The object role has the following definition in JSON. <install_dir>/json-schema/roles.json

```
 {
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "array",
  "title": "Roles",
  "description": "List of UOC roles",
  "items": {
    "type": "object",
    "title": "Role",
    "description": "Definition of a role",
    "properties": {
      "role_id": {
        "type": "string",
        "title": "Role Identifier",
```

```
        "description": "Unique identifier for the role"
      },
      "name": {
        "type": "string",
        "title": "Name",
        "description": "Presentation name of the role"
      },
      "tenant_id": {
        "type": "string",
        "title": "Tenant Identifier",
        "description": "Unique identifier of the tenant if the role is
associated to a specific tenant"
      },
      "permission": {
        "type": "array",
        "title": "Permissions",
        "description": "List of permissions associated to the role",
        "items": {
          "type": "string",
          "title": "Permission",
          "description": "Permission Identifier"
        }
      },
      "description": {
        "type": "string",
        "title": "Description",
        "description": "Descripiton of the role"
      }
    },
    "required": [ "role_id", "name", "permission" ]
  }
}
```

6.4 Permissions

The object permission has the following definition in JSON. <install_dir>/json-schema/permissions.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "array",
  "title": "Permissions",
  "description": "List of UOC permissions",
```

```
  "items": {
    "type": "object",
    "title": "Permission",
    "description": "Definition of a permission",
    "properties": {
      "id": {
        "type": "string",
        "title": "Permission Identifier",
        "description": "Unique identifier for the permission"
      },
      "name": {
        "type": "string",
        "title": "Name",
        "description": "Presentation name of the permission"
      },
      "category": {
        "type": "string",
        "title": "Category",
        "description": "Category used to organize permissions"
      },
      "operation": {
        "type": "string",
        "title": "Operation",
        "description": "Operation associated ot the permission"
      },
      "object": {
        "type": "string",
        "title": "Object",
        "description": "Object associated to the permission"
      },
      "description": {
        "type": "string",
        "title": "Description",
        "description": "Description of the permission"
      }
    },
    "required": [ "id", "name", "category", "operation", "object"  ]
  }
}
```

6.5 Workspace Categories

The object workspace category has the following definition in JSON. <install_dir>/json-schema/categories.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "array",
  "title": "Workspace Categories",
  "description": "List of workspace category",
  "items": {
    "type": "object",
    "title": "Workspace Category",
    "description": "Definition fo the workspace category",
    "properties": {
      "id": {
        "type": "string",
        "title": "Category Identifier",
        "description": "Unique identifier for the workspace category"
      },
      "tenant_id": {
        "type": "string",
        "title": "Tenant Identifier",
        "description": "Unique identifier of the tenant if the category
is associated to a specific tenant"
      },
      "name": {
        "type": "string",
        "title": "Name",
        "description": "Presentation name for the workspace category"
      },
      "roles": {
        "type": "array",
        "title": "Roles",
        "description": "List of roles associated to the category",
        "items": {
          "type": "string",
          "title": "Role",
          "description": "Role Identifier"
        }
      },
      "description": {
        "type": "string",
```

```
        "title": "Description",
        "description": "Description of the workspace category"
      },
      "icons": {
        "type": "object",
        "title": "Icons",
        "description": "Icon identifiers associated to the workspace
category",
        "properties": {
          "category": {
            "type": "string",
            "title": "Category schema.",
            "description": "Visual icon representing the workspace
category (font-awesome)"
          },
          "template": {
            "type": "string",
            "title": "Template",
            "description": "Deprecated. Please do not use"
          },
          "workspace": {
            "type": "string",
            "title": "Workspace",
            "description": "Deprecated. Please do not use"
          }
        }
      }
    },
    "required": [ "id", "name" ]
  }
}
```

6.6 Languages

The object languages has the following definition in JSON. <install_dir>/json-schema/languages.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Languages Configuration",
    "description": "Language configuration and optimization",
```

```
    "properties": {
        "optimizeL10n": {
            "type": "boolean",
            "title": "Localization Optimization",
            "description": "Prepare languages files to optimize the
initial loading",
            "default": true
        },
        "languages": {
            "type": "array",
            "title": "Languages",
            "description": "List of available languages in UOC",
            "items": {
                "type": "object",
                "title": "Language",
                "description": "",
                "properties": {
                    "name": {
                        "type": "string",
                        "title": "Name",
                        "description": "Presentation name for the
language"
                    },
                    "languageCode": {
                        "type": "string",
                        "title": "LanguageCode",
                        "description": "Unique identifier for the
language (ISO 639)"
                    },
                    "icon": {
                        "type": "string",
                        "title": "Icon",
                        "description": "Location of the icon to
associate to the language. This property is optional and by default, a
language is associated to the following icon
/public/images/languages/<languageCode>.png"
                    }
                },
                "required": [
                    "name",
                    "languageCode"
                ]
            }
```

```
        }
    },
    "required": [
        "languages"
    ]
}
```

6.7 Workspaces

The object workspace has the following definition in JSON. <install_dir>/json-schema/workspaces.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "title": "Workspaces",
    "description": "List of workspaces",
    "items": {
        "type": "object",
        "title": "Workspace",
        "description": "Definition of a workspace",
        "properties": {
            "id": {
                "type": "string",
                "title": "Workspace Identifier",
                "description": "Unique identifier of workspace"
            },
            "name": {
                "type": "string",
                "title": "Name",
                "description": "Presentation name of the workspace"
            },
            "description": {
                "type": "string",
                "title": "Description",
                "description": "Description of the workspace"
            },
            "author": {
                "type": "string",
                "title": "Author",
                "description": "Author of the workspace"
            },
```

```
            "version": {
                "type": "string",
                "title": "Version",
                "description": "Version of the workspace"
            },
            "category": {
                "type": "string",
                "title": "Category",
                "description": "Workspace category associated to the
workspace"
            },
            "domain": {
                "type": "string",
                "title": "Domain",
                "description": "Domains associated to the workspace"
            },
            "package": {
                "type": "string",
                "title": "Package",
                "description": "Package identifier associated to the
workspace",
                "default": "MAPDATA_Simulation"
            },
            "icon": {
                "type": "string",
                "title": "Icon schema.",
                "description": "Icon of the workspace
(<install_dir>/client/public/images/workspaces/<workspaceId>.png"
            },
            "roles": {
                "type": "array",
                "title": "Roles",
                "description": "List of roles associated to the
workspace",
                "items": {
                    "type": "string",
                    "title": "Role",
                    "description": "Role identifier associated to the
workspace"
                }
            },
            "favorite": {
                "type": "boolean",
```

```
                "title": "Favorite",
                "description": "Indicate this workspace is favorite and
should be displayed first in the workspace list",
                "default": false
            },
            "state": {
                "type": "string",
                "title": "State",
                "description": "State of the workspace:
UNDER_CONSTRUCTION, WAITING_VALIDATION, ACTIVE, INACTIVE"
            },
            "hideBreadcrumb": {
                "type": "boolean",
                "title": "HideBreadcrumb",
                "description": "Show or hide the internal breadcrump",
                "default": false
            },
            "hideActions": {
                "type": "boolean",
                "title": "HideActions",
                "description": "Show or hide the action button of the
workspace",
                "default": false
            },
            "navigationMode": {
                "type": "boolean",
                "title": "Navigation Mode",
                "description": "Enable or disable the navigation mode
for workspace ( one view active or multiple views in tabular windows) ",
                "default": false
            },
            "headerView": {
                "type": "object",
                "title": "Header View",
                "description": "View Identifier associated to the header
view (top of the workspace)",
                "properties": {
                    "id": {
                        "type": "string",
                        "title": "Identifier",
                        "description": "Unique Identifier of the view to
use as a header view"
                    },
```

```
                    "name": {
                        "type": "string",
                        "title": "Name",
                        "description": "Presentation Name of the header
view"
                    }
                },
                "required": [ "id", "name" ]
            },
            "footerView": {
                "type": "object",
                "title": "Footer View",
                "description": "View Identifier associated to the footer
view (bottom of the workspace)",
                "properties": {
                    "id": {
                        "type": "string",
                        "title": "Identifier",
                        "description": "Unique Identifier of the view to
use as a footer view"
                    },
                    "name": {
                        "type": "string",
                        "title": "Name",
                        "description": "Presentation Name of the footer
view"
                    }
                },
                "required": [ "id", "name" ]
            },
            "sliderView": {
                "type": "object",
                "title": "Slider View",
                "description": "View Identifier associated to the slider
view (left of the workspace)",
                "properties": {
                    "id": {
                        "type": "string",
                        "title": "Identifier",
                        "description": "Unique Identifier of the view to
use as a slider view"
                    },
                    "name": {
```

```
                        "type": "string",
                        "title": "Name",
                        "description": "Presentation Name of the slider
view"
                    },
                    "animation": {
                        "type": "string",
                        "title": "Animation",
                        "description": "Animation mode to use for the
sliding effect: fixed, push, shrink or hover",
                        "default": "fixed"
                    },
                    "width": {
                        "type": "string",
                        "title": "Width",
                        "description": "Width in pixels to use for the
slider view",
                        "default": "350px"
                    },
                    "options": {
                        "type": "object",
                        "title": "Options",
                        "description": "Options for the slider view",
                        "properties": {
                            "disableScrolling": {
                                "type": "boolean",
                                "title": "Disable Scrolling",
                                "description": "Enable or disable the
scrolling feature",
                                "default": false
                            },
                            "toggle": {
                                "type": "boolean",
                                "title": "Toggle",
                                "description": " Define the intial state
of the toggle of the slider ",
                                "default": false
                            },
                            "autohide": {
                                "type": "boolean",
                                "title": "Auto Hide",
                                "description": " Enable or disable the
automatic hide when the user clicks outside the slider view",
```

```
                                "default": false
                            }
                        }
                    }
                },
                "required": [ "id", "name" ]
            },
            "views": {
                "type": "array",
                "title": "Views",
                "description": "Definition of views associated to the
workspace",
                "items": {
                    "type": "object",
                    "title": "View",
                    "description": "Definition of the view associated to
the workspace",
                    "properties": {
                        "id": {
                            "type": "string",
                            "title": "View Identifier",
                            "description": "Unique Identifier of the
view to use in this workspace"
                        },
                        "name": {
                            "type": "string",
                            "title": "Name",
                            "description": "Presentation Name of the
view in this workspace"
                        }
                    },
                    "required": [ "id", "name" ]
                }
            },
            "$ref": "#output-parameters"
        },
        "required": [ "id", "name", "version", "views" ]
    }
}
```

6.8 Views

The object view has the following definition in JSON. <install_dir>/json-schema/views.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "title": "Views",
    "description": "List of views",
    "items": {
        "type": "object",
        "title": "View",
        "description": "Definition of the view",
        "properties": {
            "id": {
                "type": "string",
                "title": "View Identifier",
                "description": "Unique identifier of the view"
            },
            "name": {
                "type": "string",
                "title": "Name",
                "description": "Presentation name of the view"
            },
            "version": {
                "type": "string",
                "title": "Version",
                "description": "Version of the view"
            },
            "description": {
                "type": "string",
                "title": "Description",
                "description": "Description of the view"
            },
            "author": {
                "type": "string",
                "title": "Author",
                "description": "Author of the view"
            },
           "domain": {
                "type": "string",
                "title": "Domain",
                "description": "Domain associated to the view"
            },
            "package": {
```

```
                "type": "string",
                "title": "Package",
                "description": "Package associated to the view"
            },
             "roles": {
                "type": "array",
                "title": "Roles",
                "description": "List of roles associated to the view",
                "items": {
                    "type": "string",
                    "title": "Role",
                    "description": "Role identifier associated to the
view"
                }
            },
           "icon": {
                "type": "string",
                "title": "Icon",
                "description": "Icon of the layout
(<install_dir>/client/public/images/layouts/<layoutId>.png"
            },
            "layout": {
                "type": "object",
                "title": "Layout",
                "description": "Definition of the layout",
                "properties": {
                    "id": {
                        "type": "string",
                        "title": "Layout Identifier",
                        "description": "Unique identifier of the layout"
                    }
                },
                "required": [ "id" ]
            },
            "state": {
                "type": "string",
                "title": "State",
                "description": "State of the view: UNDER_CONSTRUCTION,
WAITING_VALIDATION, ACTIVE, INACTIVE"
            },
            "widgets": {
              "$ref": "#widgets"
```

```
            }
        },
        "required": [ "id", "name", "version", "layout", "widgets" ]
    }
}
```

6.9 Launch Categories

The object launch category has the following definition in JSON. <install_dir>/json-schema/launch-categories.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "title": "Launch Categories",
    "description": "List of launch category",
    "items": {
        "type": "object",
        "title": "Launch Category",
        "description": "Definition of the launch category",
        "properties": {
            "id": {
                "type": "string",
                "title": "Category Identifier",
                "description": "Unique identifier for the launch
category"
            },
            "tenant_id": {
                "type": "string",
                "title": "Tenant Identifier",
                "description": "Unique identifier of the tenant if the
launch category is associated to a specific tenant"
            },
            "name": {
                "type": "string",
                "title": "Name",
                "description": "Presentation name for the launch
category"
            },
            "icon": {
                "type": "string",
                "title": "Icon schema.",
```

```
                "description": "An explanation about the purpose of this
instance."
            },
            "description": {
                "type": "string",
                "title": "Description",
                "description": "Description of the launch category"
            }
        },
        "required": [ "id", "name" ]
    }
}
```

6.10 Launches

The object launch has the following definition in JSON. <install_dir>/json-schema/launches.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "title": "Launches",
    "description": "List of available launches",
    "items": {
        "type": "object",
        "title": "Launch",
        "description": "Definition of a launch",
        "properties": {
            "id": {
                "type": "string",
                "title": "Launch Identifier",
                "description": "Unique identifier for the launch"
            },
            "tenant_id": {
                "type": "string",
                "title": "Tenant Identifier",
                "description": "Unique identifier of the tenant if the
launch is associated to a specific tenant"
            },
            "name": {
                "type": "string",
                "title": "Name",
                "description": "Presentation name of the launch"
            },
```

```
            "description": {
                "type": "string",
                "title": "Description",
                "description": "Description of the launch"
            },
            "version": {
                "type": "string",
                "title": "Version",
                "description": "Version of the launch"
            },
            "author": {
                "type": "string",
                "title": "Author",
                "description": "Author of the launch"
            },
            "url": {
                "type": "string",
                "title": "Url",
                "description": "URL to execute for the launch"
            },
            "mode": {
                "type": "string",
                "title": "Mode",
                "description": "Mode of the execution (internal or
external): Internal mode execute the launch in a hpe-iframe widget,
external mode popup a new window or tab in the web browser"
            },
            "icon": {
                "type": "string",
                "title": "Icon",
                "description": "Iocn of the launch (font-awesome)"
            },
            "tags": {
                "type": "array",
                "title": "Launch Tags",
                "description": "Array of tags associated to the launch",
                "items": {
                    "type": "string",
                    "title": "Tag",
                    "description": "String tag to qualify the launch"
                }
            },
```

```
            "category": {
                "type": "string",
                "title": "Category",
                "description": "Category of the launch. This category is
used in menu for organizing the launch display"
            },
            "state": {
                "type": "string",
                "title": "State",
                "description": "State of the launch: UNDER_CONSTRUCTION,
WAITING_VALIDATION, ACTIVE, INACTIVE"
            }
        },
        "required": [ "id", "name", "version", "author", "url", "mode",
"tags", "state" ]
    }
}
```

6.11 Notification Composite Key

The object notification has the following definition in JSON. The composite key refers to a notification. <install_dir>/json-schema/notification-composite-key.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Notification Composite Key",
    "description": "An explanation about the purpose of this instance.",
    "properties": {
        "type": {
            "type": "string",
            "title": "Notification Type",
            "description": "Type of the notification (e.g. alert or
data)"
        },
        "id": {
            "type": "string",
            "title": "Notification Identifier",
            "description": "Identifier of the notification"
        },
        "origin": {
            "type": "string",
            "title": "Origin",
```

```
            "description": "Origin of the notification (server, plugin,
client or workspace)"
        },
        "domain": {
            "type": "string",
            "title": "Domain",
            "description": "Domain identifier where the notification is
coming from"
        },
        "package": {
            "type": "string",
            "title": "Package",
            "description": "Package identifier where the notification is
coming from"
        },
        "workspace": {
            "type": "string",
            "title": "Workspace",
            "description": "Workspace identifier where the notification
is coming from"
        }
    },
    "required": [ "type", "id","origin" ]
}
```

6.12 Notifications

The object notification has the following definition in JSON. <install_dir>/json-schema/notifications.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "title": "Notifications",
    "description": "List of notifications",
    "items": {
        "type": "object",
        "title": "Notification",
        "description": "Definition of the notification",
        "properties": {
            "$ref": "#notification-composite-key",
            "roles": {
                "type": "array",
                "title": "Roles",
```

```
                "description": "List of roles associated to the
notificayion",
                "items": {
                    "type": "string",
                    "title": "Role",
                    "description": "Role identifier associated to the
notification"
                }
            },
            "level": {
                "type": "string",
                "title": "Level",
                "description": "Level of the notification (error,
warning, information...)"
            },
            "update": {
                "type": "boolean",
                "title": "Update",
                "description": "Indicate if the notification is an
update and should replace a previous one"
            },
            "title": {
                "type": "string",
                "title": "Title",
                "description": "title to display for the notification"
            },
            "data": {
                "type": "object",
                "title": "Data",
                "description": "Data of the notification",
                "properties": {
                    "text": {
                        "type": "object",
                        "title": "Text",
                        "description": "Description of the notification
in multiple locales",
                        "properties": {
                            "en-us": {
                                "type": "string",
                                "title": "Message of the notification
for this locale",
                                "description": "Detailed description of
the notification"
                            }
```

```
                        }
                    },
                    "occurences": {
                        "type": "integer",
                        "title": "Occurences",
                        "description": "Number of occurrences of the
notification received at server side. it will be displayed as an
information"
                    }
                }
            },
            "actions": {
                "$ref": "#actions"
            }
        },
        "icon": {
            "type": "object",
            "title": "Icon",
            "description": "By default, each level has a pre-defined
icon, but it is possible to specify a custom icon from
(<install_dir>/client/public/images/notifications/<notificationId>.png",
            "properties": {
                "class": {
                    "type": "string",
                    "title": "Class",
                    "description": "Icon defined as a class. Support for
font-awesome"
                }
            }
        },
        "sound": {
            "type": "object",
            "title": "Sound",
            "description": "Sound associated to this notification",
            "properties": {
                "file": {
                    "type": "string",
                    "title": "File",
                    "description": "By default, each level has a pre-
defined sound, but it is possible to specify a custom sound from
(<install_dir>/client/public/sounds/notifications/<notificationId>.mp3"
                }
            }
        },
```

```
        "keywords": {
            "type": "array",
            "title": "Keywords",
            "description": "list of keyword used to classify the
notification and ease analysis",
            "items": {
                "type": "string",
                "title": "Keyword",
                "description": "Keyword associated to this notification"
            }
        },
        "display": {
            "type": "object",
            "title": "Display",
            "description": "Options to customize the display of the
notification",
            "properties": {
                "type": {
                    "type": "string",
                    "title": "Type",
                    "description": "Graphical effect: toast, banner,
popup"
                },
                "options": {
                    "type": "object",
                    "title": "Options",
                    "description": "Advanced options",
                    "properties": {
                        "timeout": {
                            "type": "integer",
                            "title": "Timeout",
                            "description": "Timeout in milliseconds of
the notification before closing automatically the notification",
                            "default": 5000
                        },
                        "dismissible": {
                            "type": "boolean",
                            "title": "Dismissible",
                            "description": "Show or hide the close
action. Some notificaiton cannot be closed until a user selects an
action from the recommanded action list"
                        }
                    }
```

```
                },
                "required": [ "origin", "type", "id" ]
            }
        }
    }
}
```

6.13 Plugins

The object plugin has the following definition in JSON. <install_dir>/json-schema/plugin.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Plugin",
    "description": "Definition of a plugin",
    "properties": {
        "id": {
            "type": "string",
            "title": "Plugin Identifier",
            "description": "Unique identifier for the plugin. This
identifier must be validated and registered with the support team to be
sure it is unique."
        },
        "name": {
            "type": "string",
            "title": "Name",
            "description": "Presentation name of the plugin"
        },
        "description": {
            "type": "string",
            "title": "Description",
            "description": "Description of the plugin"
        },
        "version": {
            "type": "string",
            "title": "Version",
            "description": "Version of the plugin"
        },
        "author": {
            "type": "string",
            "title": "Author",
            "description": "Author of the plugin"
```

```
        },
        "notifications": {
            "type": "array",
            "title": "Notifications",
            "description": "List of notifications that this plugin can
send",
            "items": { "$ref": "#notification" }
        }
    },
    "required": [ "id", "name", "version" ]
}
```

IMPORTANT: Plugin Id must match the Add-on Plugin ID Registration to avoid any conflict in production when many other products have been installed on the same platform

6.14 Widgets

The object widget has the following definition in JSON. <install_dir>/json-schema/widgets.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "title": "Widgets",
    "description": "Definition of widgets",
    "items": {
        "type": "object",
        "title": "Widget",
        "description": "Definition of a widget",
        "properties": {
            "widgetId": {
                "type": "string",
                "title": "Widget Identifier",
                "description": "Unique identifier of the widget"
            },
            "name": {
                "type": "string",
                "title": "Name",
                "description": "Presentation name of the widget"
            },
            "description": {
```

```
                "type": "string",
                "title": "Description",
                "description": "Description of the widget"
            },
            "author": {
                "type": "string",
                "title": "Author",
                "description": "Author of the widget"
            },
            "version": {
                "type": "string",
                "title": "Version",
                "description": "Version of the widget"
            },
            "icon": {
                "type": "string",
                "title": "Icon",
                "description": "Icon of the widget
(<install_dir>/client/public/images/widgets/<widgetId>.png"
            },
            "categories": {
                "type": "array",
                "title": "Categories",
                "description": "Categories associated to the widget",
                "items": {
                    "type": "string",
                    "title": "Category",
                    "description": "Category associated to the widget"
                }
            },
            "domains": {
                "type": "array",
                "title": "Domains",
                "description": "List of domains associated to the
widget",
                "items": {
                    "type": "string",
                    "title": "Domain Identifier",
                    "description": "Domain identifier associated to the
widget"
                }
            },
```

```
            "packages": {
                "type": "array",
                "title": "Packages",
                "description": "List of packages associated to the
widget",
                "items": {
                    "type": "string",
                    "title": "Package Identifier",
                    "description": "Package identifier associated to the
widget"
                }
            },
            "templateUrl": {
                "type": "string",
                "title": "Template Url",
                "description": "HTML template associated to this widget"
            },
            "optionsTemplateUrl": {
                "type": "string",
                "title": "OptionsTemplateUrl schema.",
                "description": "HTML template associated to this widget
for the configuration panel"
            },
            "toolbar": {
                "type": "object",
                "title": "Widget Toolbar",
                "description": "Definition of the widget toolbar",
                "properties": {
                    "zoom": {
                        "type": "boolean",
                        "title": "Full Page",
                        "description": "Show or hide the full page
button"
                    },
                    "configuration": {
                        "type": "boolean",
                        "title": "Configuration",
                        "description": "Show or hide the configure
button"
                    },
                    "refresh": {
                        "type": "boolean",
                        "title": "Refresh",
```

```
                    "description": "Show or hide the refresh button"
                },
                "selectors": {
                    "type": "boolean",
                    "title": "Data Selection",
                    "description": "Show or hide the select data
button"
                }
            }
        },
        "$ref": "#input-parameters",
        "$ref": "#output-parameters"
    },
    "required": [ "widgetId", "name", "version", "templateUrl" ]
  }
}
```

6.15 Layouts

The object layout has the following definition in JSON. <install_dir>/json-schema/layouts.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "title": "Layouts",
    "description": "List of layouts",
    "items": {
        "type": "object",
        "title": "Layout",
        "description": "Definition of a layout",
        "properties": {
            "layoutId": {
                "type": "string",
                "title": "Layout Identifier",
                "description": "Unique identifier for the layout"
            },
            "name": {
                "type": "string",
                "title": "Name",
                "description": "Presentation name of the layout"
            },
```

```
            "description": {
                "type": "string",
                "title": "Description",
                "description": "Description of the layout"
            },
            "author": {
                "type": "string",
                "title": "Author",
                "description": "Author of the layout"
            },
            "version": {
                "type": "string",
                "title": "Version",
                "description": "Version of the layout"
            },
            "icon": {
                "type": "string",
                "title": "Icon",
                "description": "icon of the layout
(<install_dir>/client/public/images/layouts/<layoutId>.png"
            },
            "categories": {
                "type": "array",
                "title": "Categories",
                "description": "An explanation about the purpose of this
instance.",
                "items": {
                    "type": "string",
                    "title": "Category",
                    "description": "Category associated to the layout"
                }
            },
            "domains": {
                "type": "array",
                "title": "Domains",
                "description": "List of domains associated to the
layout",
                "items": {
                    "type": "string",
                    "title": "Domain Identifier",
                    "description": "Domain identifier associated to the
layout"
                }
```

```
            },
            "packages": {
                "type": "array",
                "title": "Packages",
                "description": "List of packages associated to the
layout",
                "items": {
                    "type": "string",
                    "title": "Package Identifier",
                    "description": "Package identifier associated to the
layout"
                }
            },
            "widgetCount": {
                "type": "integer",
                "title": "Widget Count",
                "description": "Maximal number of widget that this
layout can support. This count is used by the view designer to recommend
you a list of layout that can fit your need."
            },
            "responsive": {
                "type": "boolean",
                "title": "Responsive",
                "description": "Boolean to indicate if the layout has
been built to support responsive web design"
            },
            "templateUrl": {
                "type": "string",
                "title": "Template Url",
                "description": "HTML template associated to this layout"
            }
        },
        "required": [ "layoutId", "name", "version", "widgetCount",
"responsive", "templateUrl" ]
    }
}
```

6.16 Menu Bars

The object menu bar has the following definition in JSON. <install_dir>/json-schema/menu-bars.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
```

```
  "type": "array",
  "title": "Menu Bars",
  "description": "List of Menu Bars",
  "items": {
    "type": "object",
    "title": "Menu Bar",
    "description": "Definition of a menu bar",
    "properties": {
      "menuBarId": {
        "type": "string",
        "title": "Menu Bar Identifier",
        "description": "Unique identifier for the menu bar"
      },
      "name": {
        "type": "string",
        "title": "Name",
        "description": "Presentation name of the menu bar"
      },
      "description": {
        "type": "string",
        "title": "Description",
        "description": "Description of the menu bar"
      },
      "author": {
        "type": "string",
        "title": "Author",
        "description": "Author of the menu bar"
      },
      "version": {
        "type": "string",
        "title": "Version",
        "description": "Version of the menu bar"
      },
      "icon": {
        "type": "string",
        "title": "Icon",
        "description": "icon of the layout
(<install_dir>/client/public/images/menu-bars/<menuBarId>.png"
      },
      "templateUrl": {
        "type": "string",
        "title": "Template Url",
```

```
        "description": "HTML template associated to this menu bar"
      }
    },
    "required": [ "menuBarId", "name", "version", "templateUrl" ]
  }
}
```

6.17 Menu Items

The object menu item has the following definition in JSON. <install_dir>/json-schema/menu-items.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "array",
  "title": "Menu items",
  "description": "List of Menu Items",
  "items": {
    "type": "object",
    "title": "Menu Item",
    "description": "Definition of Menu Item",
    "properties": {
      "menuItemId": {
        "type": "string",
        "title": "Menu Item Identifier",
        "description": "Unique identifier for the menu item"
      },
      "name": {
        "type": "string",
        "title": "Name",
        "description": "Presentation name of the menu item"
      },
      "description": {
        "type": "string",
        "title": "Description",
        "description": "Description of the menu item"
      },
      "author": {
        "type": "string",
        "title": "Author",
        "description": "Author of the menu item"
      },
```

```
      "version": {
        "type": "string",
        "title": "Version",
        "description": "Version of the menu item"
      },
      "icon": {
        "type": "string",
        "title": "Icon",
        "description": "Icon of the menu item
(<install_dir>/client/public/images/menu-items/<menuItemId>.png"
      },
      "templateUrl": {
        "type": "string",
        "title": "Template Url",
        "description": "HTML template associated to this menu item"
      }
    },
    "required": [ "menuItemId", "name", "version", "templateUrl" ]
  }
}
```

6.18 Themes

The object theme has the following definition in JSON. <install_dir>/json-schema/themes.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "array",
  "title": "Themes",
  "description": "List of Themes",
  "items": {
    "type": "object",
    "title": "Theme",
    "description": "Definition of a Theme",
    "properties": {
      "themeId": {
        "type": "string",
        "title": "Theme Identifier",
        "description": "Unique identifier for the theme"
      },
      "name": {
```

```
        "type": "string",
        "title": "Name",
        "description": "Presentation name of theme"
      },
      "description": {
        "type": "string",
        "title": "Description",
        "description": "Description of the theme"
      },
      "author": {
        "type": "string",
        "title": "Author",
        "description": "Author of the theme"
      },
      "version": {
        "type": "string",
        "title": "Version",
        "description": "Version of the theme"
      },
      "icon": {
        "type": "string",
        "title": "Icon",
        "description": "Icon of the theme
(<install_dir>/client/public/images/themes/<themeId>.png"
      },
      "location": {
        "type": "string",
        "title": "Location",
        "description": "Relative directory where to find the full
implementation of the theme. It should be
/addons/<addonId>/themes/<themeId>"
      }
    },
    "required": [ "themeId", "name", "version", "location" ]
  }
}
```

6.19 Modules

The object module has the following definition in JSON. **<install_dir>/json-schema/modules.json**

```
{
```

```
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "array",
    "title": "Modules",
    "description": "List of Modules",
    "items": {
        "type": "object",
        "title": "Module",
        "description": "Definition of a Module",
        "properties": {
            "id": {
                "type": "string",
                "title": "Module Identifier",
                "description": "Unique identifier for the module"
            },
            "name": {
                "type": "string",
                "title": "Name",
                "description": "Presentation name of the module"
            },
            "type": {
                "type": "string",
                "title": "Type",
                "description": "Type of module to quality better the
extension (e.g. filter, service, schemaform...)"
            },
            "description": {
                "type": "string",
                "title": "Description",
                "description": "Description of the module"
            },
            "author": {
                "type": "string",
                "title": "Author",
                "description": "Author of the module"
            },
            "version": {
                "type": "string",
                "title": "Version",
                "description": "Version of the module"
            },
            "icon": {
                "type": "string",
```

```
                "title": "Icon",
                "description": "Icon of the module
(<install_dir>/client/public/images/modules/<moduleId>.png"
            },
            "location": {
                "type": "string",
                "title": "Location",
                "description": "Relative directory where to find the
implementation of the module. It should be
/addons/<addonId>/modules/<moduleId>"
            }
        },
        "required": [
            "id",
            "name",
            "version",
            "location"
        ]
    }
}
```

6.20 Launch Keywords

The object launch keyword has the following definition in JSON. <install_dir>/json-schema/modules.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "title": "Launch Keyword",
  "description": "Definition of Launch Keyword",
  "properties": {
    "id": {
      "type": "string",
      "title": "Launch Keyword Identifier",
      "description": "Unique identifier for the launch keyword"
    },
    "name": {
      "type": "string",
      "title": "Name",
      "description": "Presentation name of the launch keyword"
    },
    "description": {
```

```
      "type": "string",
      "title": "Description",
      "description": "Description of the launch keyword"
    },
    "author": {
      "type": "string",
      "title": "Author",
      "description": "Author of the launch keyword"
    },
    "version": {
      "type": "string",
      "title": "Version",
      "description": "Version of the launch keyword"
    },
    "keywords": {
      "type": "array",
      "title": "Keywords",
      "description": "Definition of keywords for the launch keyword",
      "items": {
        "type": "object",
        "title": "Keyword",
        "description": "Definition of the keyword",
        "properties": {
          "group": {
            "type": "string",
            "title": "Group",
            "description": "Group of the keyword. it will be the prefix
of the keyword e.g. <group:name>"
          },
          "name": {
            "type": "string",
            "title": "Name",
            "description": "Presentation name of the keyword.it will be
the postfix of the keyword e.g. <group:name>"
          },
          "description": {
            "type": "string",
            "title": "Description",
            "description": "Description of the keyword"
          }
        },
        "required": [ "group", "name" ]
```

```
      }
    }
  },
  "required": [ "id", "name", "version", "keywords" ]
}
```

6.21 Object Type

The object type definition is very flexible and must follow the JSON Schema standard.

JSON Schema is a vocabulary that allows you describe the existing data format and ease the validation of JSON documents. The latest IETF published draft is v4.

NOTE: Please refer to JSON schema specification and JSON schema (http://json-schema.org)

6.22 Input Parameters

The object input parameter involved in the communication between uoc components has the following definition in JSON. **<install_dir>/json-schema/input-parameters.json**

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "inputs": {
            "type": "array",
            "title": "Input Parameters",
            "description": "Definition of input parameters",
            "items": {
                "type": "object",
                "title": "Input Parameter",
                "description": "Definition of an input parameter",
                "properties": {
                    "id": {
                        "type": "string",
                        "title": "Input Parameter Identifier",
                        "description": "Unique identifier of the input
parameter"
                    },
                    "description": {
                        "type": "string",
                        "title": "Description",
```

```
                        "description": "Description of the input
parameter"
                    },
                    "name": {
                        "type": "string",
                        "title": "Name",
                        "description": "Presentation name of the input
parameter"
                    },
                    "defaultValue": {
                        "type": "string",
                        "title": "Default Value",
                        "description": "Value set to the input parameter
by default. it could be a string, number, date, json..."
                    },
                    "listen": {
                        "type": "boolean",
                        "title": "Listen",
                        "description": "Listen value changes of the
input parameter. If set to true, a event is sent to the component
listeing changes",
                        "default": false
                    },
                    "required": {
                        "type": "boolean",
                        "title": "Required",
                        "description": "Indicate that this input
parameter is mandatory (filled) before performing any request tio the
data server",
                        "default": false
                    },
                    "edit": {
                        "type": "boolean",
                        "title": "Edit",
                        "description": "Indicate to the View Designer
that this input parameter is read-only and cannot be edited",
                        "default": false
                    }
                },
                "required": [ "id", "name" ]
            }
        }
    }
}
```

6.23 Output Parameters

The object input parameter involved in the communication between uoc components has the following definition in JSON. <install_dir>/json-schema/output-parameters.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "outputs": {
            "type": "array",
            "title": "Output Parameters",
            "description": "Definition of output parameters",
            "items": {
                "type": "object",
                "title": "Output Parameter",
                "description": "Definition of an output parameter",
                "properties": {
                    "name": {
                        "type": "string",
                        "title": "Name",
                        "description": "Presentation name of the output
parameter"
                    },
                    "id": {
                        "type": "string",
                        "title": "Output Parameter Identifier",
                        "description": "Unique identifier of the output
parameter"
                    },
                    "defaultValue": {
                        "type": "string",
                        "title": "Default Value",
                        "description": "Value set to the input parameter
by default. it could be a string, number, date, json..."
                    }
                },
                "required": [ "name", "id" ]
            }
        }
    }
}
```

6.24 Breadcrumb Definition

The object breadcrumb definition is mandatory for hpe-breadcrumb widget. It feeds the input parameter BREADCRUMB_DEF. It has the following definition in JSON. <install_dir>/json-schema/breadcrumb-definition.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "title": "Breadcrumb definition",
  "description": "Breadcrumb definition is defined inside the
defaultValue of an output of a view",
  "properties": {
    "defaultValue": {
      "type": "object",
      "title": "DefaultValue",
      "description": "DefaultValue property of the output parameter",
      "properties": {
        "separator": {
          "type": "string",
          "title": "Separator",
          "description": "The separator between 2 breadcrumbItems",
          "default": "/"
        },
        "breadcrumbItems": {
          "type": "array",
          "title": "BreadcrumbItems",
          "description": "Set of breadcrumbItems",
          "items": {
            "type": "object",
            "title": "breadcrumbItem",
            "description": "A breadcrumb item contains an icon, a title
and a link, all non-mandatory",
            "properties": {
              "icon": {
                "type": "string",
                "title": "Icon",
                "description": "Breadcrumb item icon. example: 'fa fa-
home'"
              },
              "title": {
                "type": "string",
                "title": "Title",
                "description": "Breadcrumb item title"
```

```
              },
              "link": {
                "type": "string",
                "title": "Link",
                "description": "The url the user will be redirected when
he clicks on the breadcrumb item. The link applies on icon and title.
example: 'workspace-manager/Demo'"
              }
            }
          }
        }
      }
    }
  }
}
```

6.25 Widget Selections

The object widget selection (Identifier is WIDGET_SELECTIONS) parameter involved in the communication between uoc components has the following definition in JSON.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Widget selection json schema",
    "description": "Widget selection json schema",
    "properties": {
        "widgetSelections": {
            "type": "object",
            "title": "WidgetSelections",
            "description": "The widgetSelections describes the data
selection at the widget level. It is used for the communication between
uoc components (Identifier is WIDGET_SELECTIONS)",
            "properties": {
                "dataSelection": {
                    "$ref": "#dataSelection"
                },
                "dataFilters": {
                    "$ref": "#dataFilters"
                },
                "topFilter": {
                    "$ref": "#topFilter"
```

```
                }
            }
        }
    }
}
```

Example of widget selection collecting all DNS requests for brand equal to “Huawei” and technology equal to “3G”

```
[{
    "dataSelection": {
        "factSelection": [{
            "domain": "ossa",
            "package": "MBBQOE_Trial",
            "dimensions": [{
                "id": "BRAND", "folder": ["DEVICE", "Device"]
            }, {
                "id": "TECHNOLOGY", "folder": ["NETWORK", "Cell"]
            }],
            "facts": [{
                "id": "DNS_req_count", "folder": ["DNS", "DNS
Accessibility"]
            }]}]
    },
    "dataFilters": {
        "filters": [{
            "domain": "ossa",
            "package": "MBBQOE_Trial",
            "dimensionFilters": [{
                "dimensionId": "BRAND", "eq": ["Huawei"] }, {
"dimensionId": "TECHNOLOGY","eq": ["3G"] }]
        }]    }
}]
```

6.26 Data Selection

The object data selection involved in the data request with dimensions / facts format to the data server has the following definition in JSON.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Data selection json schema",
    "description": "Data selection json schema",
```

```
    "properties": {
        "dataSelection": {
            "type": "object",
            "title": "DataSelection",
            "description": "The dataSelection describes what kind of
data is selected. It can be a fact selection or a type selection coupled
with a form selection",
            "oneOf": [
                {
                    "properties": {
                        "typeSelection": {
                            "$ref": "#typeSelection"
                        },
                        "formSelection": {
                            "$ref": "#formSelection"
                        }
                    },
                    "required": [
                        "typeSelection"
                    ]
                },
                {
                    "properties": {
                        "factSelection": {
                            "$ref": "#factSelection"
                        }
                    },
                    "required": [
                        "factSelection"
                    ]
                }
            ]
        }
    },
    "required": [
        "dataSelection"
    ]
}
```

6.27 Fact Selection

The object data selection involved in the data request with dimensions / facts format to the data server has the following definition in JSON. <install_dir>/json-schema/fact-selection.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "factSelection": {
            "type": "array",
            "title": "Fact Selections",
            "description": "List of fact selections, e.g. Dimensions /
Facts requests",
            "items": {
                "type": "object",
                "title": "Fact Selection",
                "description": "Definition of a fact selection",
                "properties": {
                    "domain": {
                        "type": "string",
                        "title": "Domain Identifier",
                        "description": "Domain Identifier associated to
the fact selection"
                    },
                    "package": {
                        "type": "string",
                        "title": "Package Identifier",
                        "description": "Package Identifier associated to
the fact selection"
                    },
                    "dimensions": {
                        "type": "array",
                        "title": "Dimensions",
                        "description": "Definition of the dimensions to
use for the fact selection",
                        "items": {
                            "type": "object",
                            "title": "Dimension",
                            "description": "Dimension to use for the
fact selection",
                            "properties": {
                                "id": {
                                    "type": "string",
```

```
                                    "title": "Dimension Identifier",
                                    "description": "Dimension Identifier
to use for the fact selection"
                                },
                                "folder": {
                                    "type": "array",
                                    "title": "Folders",
                                    "description": "Hierarchical folders
to identify the dimension",
                                    "items": {
                                        "type": "string",
                                        "title": "Folder",
                                        "description": "Dimension Folder
to use for the fact selection"
                                    }
                                }
                            },
                            "required": ["id", "folder" ]
                        }
                    },
                    "facts": {
                        "type": "array",
                        "title": "Facts",
                        "description": "Definition of the facts linked
to dimensions to use for the fact selection",
                        "items": {
                            "type": "object",
                            "title": "Fact",
                            "description": "Definition of the facts to
use for the fact selection",
                            "properties": {
                                "id": {
                                    "type": "string",
                                    "title": "Identifier",
                                    "description": "Fact Identifier to
use for the fact selection"
                                },
                                "folder": {
                                    "type": "array",
                                    "title": "Folders",
                                    "description": "Hierarchical folders
to identify the fact",
                                    "items": {
                                        "type": "string",
```

```
                                        "title": "Folder",
                                        "description": "Fact Folder to
use for the fact selection"
                                    }
                                }
                            },
                            "required": [ "id", "folder" ]
                        }
                    }
                },
                "required": ["domain", "package", "dimensions", "facts"]
            }
        }
    },
    "required": [ "factSelection" ]
}
```

6.28 Type Selection

The object type selection involved in the data request with the object type format to the data server has the following definition in JSON. <install_dir>/json-schema/type-selection.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "typeSelection": {
      "type": "object",
      "title": "Type Selection",
      "description": "Definition of a Object Type Selection",
      "properties": {
        "domain": {
          "type": "string",
          "title": "Domain Identifier",
          "description": "Domain Identifier associated to the object"
        },
        "package": {
          "type": "string",
          "title": "Package Identifier",
          "description": "Package Identifier associated to the object"
        },
```

```
        "type": {
          "type": "string",
          "title": "Object Type Identifier",
          "description": "Object Type Identifier to use for the
selection"
        },
        "version": {
          "type": "string",
          "title": "Version",
          "description": "Version of the Object Type to use for the
selection"
        }
      },
      "required": [ "domain", "package", "type" ]
    }
  },
  "required": [
    "typeSelection"
  ]
}
```

6.29 Form Selection

The form selection involved in the form popup widget. <install_dir>/json-schema/form-selection.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "formSelection": {
      "type": "object",
      "title": "Form Selection",
      "description": "Definition of the Form Selection",
      "properties": {
        "id": {
          "type": "string",
          "title": "Form Identifier",
          "description": "Form identifier to use for the selection"
        },
        "version": {
          "type": "string",
          "title": "Version schema.",
```

```
          "description": "Version of the form to use for the selection"
        }
      },
      "required": [ "id" ]
    }
  },
  "required": [ "formSelection" ]
}
```

6.30 Top Filter

The object top filter involved in the data request to the data server has the following definition in JSON.
<install_dir>/json-schema/top-filter.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "topFilter": {
      "type": "object",
      "title": "Top Filter",
      "description": "Definition of a top filter",
      "properties": {
        "topValue": {
          "type": "integer",
          "title": "Top Value",
          "description": "Number of top value to retrieve in the query"
        },
        "worstToBest": {
          "type": "boolean",
          "title": "Worst To Best",
          "description": "Top of worst or best value. (ascending /
descending sorting based on the metric)",
          "default": false
        }
      },
      "required": [ "topValue" ]
    }
  },
  "required": [ "topFilter" ]
}
```

6.31 Top Filter Configuration

The object top filter configuration has the following definition in JSON. <install_dir>/json-schema/top-filter-configuration.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "topFilterConfiguration": {
      "type": "object",
      "title": "Top Filter Configuration",
      "description": "Configuration of the Top Filter",
      "properties": {
        "minTopValue": {
          "type": "integer",
          "title": "Minimal Top Value",
          "description": "Minimal limit used for the top value",
          "default": 0
        },
        "maxTopValue": {
          "type": "integer",
          "title": "MaxTopValue schema.",
          "description": "Maximal limit used for the top value",
          "default": 20
        },
        "stepTopValue": {
          "type": "integer",
          "title": "Step Top Value",
          "description": "Minimal value to use for each step of the top
value",
          "default": 1
        }
      }
    }
  },
  "required": [
    "topFilterConfiguration"
  ]
}
```

6.32 Data Filters

The object data filter involved in the data request with dimensions / facts format to the data server has the following definition in JSON. <install_dir>/json-schema/data-filters.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "title": "Data filters",
  "description": "Definition of the Data filters",
  "properties": {
    "dataFilters": {
      "type": "object",
      "title": "Data Filters",
      "description": "Definition of Data Filters",
      "oneOf": [
        {
          "properties": {
            "filters": {
              "type": "array",
              "title": "Filters",
              "description": "List of data filters",
              "items": {
                "type": "object",
                "title": "Filter",
                "description": "Definition of a data filter",
                "properties": {
                  "domain": {
                    "type": "string",
                    "title": "Domain Identifier",
                    "description": "Domain Identifier associated to the
filters"
                  },
                  "package": {
                    "type": "string",
                    "title": "Package Identifier",
                    "description": "Package Identifier associated to the
filters"
                  },
                  "dimensionFilters": {
                    "$ref": "#dimensionFilters"
```

```
                }
              },
              "required": [
                "domain",
                "package"
              ]
            }
          }
        },
        "required": [
          "filters"
        ]
      },
      {
        "properties": {
          "objectFilters": {
            "$ref": "#objectFilters"
          }
        },
        "required": [
          "objectFilters"
        ]
      }
    ]
  }
}
```

6.33 Dimension Filters

The object dimension filters configuration has the following definition in JSON. <install_dir>/json-schema/dimension-filters.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Dimension filters",
    "description": "Definition of the dimension filters",
    "properties": {
        "dimensionFilters": {
            "type": "array",
```

```
            "title": "Dimension Filters",
            "description": "List of dimension filters associated to the
data filters",
            "items": {
                "type": "object",
                "title": "Filter",
                "description": "Definition of a dimension filter",
                "properties": {
                    "dimensionId": {
                        "type": "string",
                        "title": "Dimension Identifier",
                        "description": "Dimension identifier on which
the filter will be applied"
                    },
                    "eq": {
                        "type": "array",
                        "title": "Operator Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "startsWith": {
                        "type": "array",
                        "title": "Operator Starts With",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "contains": {
                        "type": "array",
                        "title": "Operator Contains",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
```

```
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "endsWith": {
                        "type": "array",
                        "title": "Operator Ends With",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "match": {
                        "type": "array",
                        "title": "Operator Match",
                        "description": "Filter value to apply for the
operator(regexp). (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "gt": {
                        "type": "array",
                        "title": "Operator Greater Than",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "ge": {
                        "type": "array",
```

```
                        "title": "Operator Greather Than or Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "lt": {
                        "type": "array",
                        "title": "Operator Less Than",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "le": {
                        "type": "array",
                        "title": "Operator Less Than or Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "isnull": {
                        "type": "array",
                        "title": "Operator Is Null",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
```

```
                        }
                    },
                    "not_eq": {
                        "type": "array",
                        "title": "Operator Not Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_startsWith": {
                        "type": "array",
                        "title": "Operator Not Starts With",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_contains": {
                        "type": "array",
                        "title": "Operator Not Contains",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_endsWith": {
                        "type": "array",
                        "title": "Operator Not Ends with",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
```

```
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_match": {
                        "type": "array",
                        "title": "Operator Not Match",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_gt": {
                        "type": "array",
                        "title": "Operator Not Greater Than",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_ge": {
                        "type": "array",
                        "title": "Operator Not Greater Than or Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_lt": {
                        "type": "array",
```

```
                        "title": "Operator Not Less Than",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_le": {
                        "type": "array",
                        "title": "Operator not Less Than or Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_isnull": {
                        "type": "array",
                        "title": "Operator Not Null",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    }
                },
                "required": [
                    "dimensionId"
                ]
            }
        }
    }
}
```

6.34 Object Filters

The object object filters configuration has the following definition in JSON. <install_dir>/json-schema/object-filters.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Object filters",
    "description": "Definition of the object filters",
    "properties": {
        "objectFilters": {
            "type": "array",
            "title": "object Filters",
            "description": "List of object filters associated to the
data filters",
            "items": {
                "type": "object",
                "title": "Filter",
                "description": "Definition of an object filter",
                "properties": {
                    "propertyId": {
                        "type": "string",
                        "title": "propertyId",
                        "description": "The object's property identifier
on which the filter will be applied"
                    },
                    "eq": {
                        "type": "array",
                        "title": "Operator Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "startsWith": {
                        "type": "array",
                        "title": "Operator Starts With",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
```

```
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "contains": {
                        "type": "array",
                        "title": "Operator Contains",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "endsWith": {
                        "type": "array",
                        "title": "Operator Ends With",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "match": {
                        "type": "array",
                        "title": "Operator Match",
                        "description": "Filter value to apply for the
operator(regexp). (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "gt": {
```

```
                        "type": "array",
                        "title": "Operator Greater Than",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "ge": {
                        "type": "array",
                        "title": "Operator Greather Than or Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "lt": {
                        "type": "array",
                        "title": "Operator Less Than",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "le": {
                        "type": "array",
                        "title": "Operator Less Than or Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
```

```
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "isnull": {
                        "type": "array",
                        "title": "Operator Is Null",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_eq": {
                        "type": "array",
                        "title": "Operator Not Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_startsWith": {
                        "type": "array",
                        "title": "Operator Not Starts With",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_contains": {
                        "type": "array",
                        "title": "Operator Not Contains",
```

```
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_endsWith": {
                        "type": "array",
                        "title": "Operator Not Ends with",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_match": {
                        "type": "array",
                        "title": "Operator Not Match",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_gt": {
                        "type": "array",
                        "title": "Operator Not Greater Than",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
```

```
                    },
                    "not_ge": {
                        "type": "array",
                        "title": "Operator Not Greater Than or Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_lt": {
                        "type": "array",
                        "title": "Operator Not Less Than",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_le": {
                        "type": "array",
                        "title": "Operator not Less Than or Equal",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    },
                    "not_isnull": {
                        "type": "array",
                        "title": "Operator Not Null",
                        "description": "Filter value to apply for the
operator. (boolean OR is applied in case of multiple values)",
                        "items": {
                            "type": "string",
```

```
                            "title": "Value",
                            "description": "Value to apply for the
filter"
                        }
                    }
                },
                "required": [
                    "propertyId"
                ]
            }
        }
    }
}
```

6.35 Packages Configuration

The object packages configuration has the following definition in JSON. <install_dir>/json-schema/packages-configuration.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Packages Configuration",
    "description": "List of packages visible for analysis",
    "properties": {
        "packagesConfiguration": {
            "type": "array",
            "title": "PackagesConfiguration",
            "description": "Definition of the configuration for
packages",
            "items": {
                "type": "object",
                "title": "Fact Selection",
                "description": "Definition of a fact selection",
                "properties": {
                    "domain": {
                        "type": "string",
                        "title": "Domain Identifier",
                        "description": "Domain Identifier associated to
the fact selection"
                    },
                    "package": {
```

```
                        "type": "string",
                        "title": "Package Identifier",
                        "description": "Package Identifier associated to
the fact selection"
                    },
                    "dimensions": {
                        "type": "array",
                        "title": "Dimensions",
                        "description": "Definition of the dimensions to
use for the fact selection",
                        "items": {
                            "type": "object",
                            "title": "Dimension",
                            "description": "Dimension to use for the
fact selection",
                            "properties": {
                                "id": {
                                    "type": "string",
                                    "title": "Dimension Identifier",
                                    "description": "Dimension Identifier
to use for the fact selection"
                                },
                                "folder": {
                                    "type": "array",
                                    "title": "Folders",
                                    "description": "Hierarchical folders
to identify the dimension",
                                    "items": {
                                        "type": "string",
                                        "title": "Folder",
                                        "description": "Dimension Folder
to use for the fact selection"
                                    }
                                }
                            },
                            "required": [
                                "id",
                                "folder"
                            ]
                        }
                    },
                    "facts": {
                        "type": "array",
```

```
                        "title": "Facts",
                        "description": "Definition of the facts linked
to dimensions to use for the fact selection",
                        "items": {
                            "type": "object",
                            "title": "Fact",
                            "description": "Definition of the facts to
use for the fact selection",
                            "properties": {
                                "id": {
                                    "type": "string",
                                    "title": "Identifier",
                                    "description": "Fact Identifier to
use for the fact selection"
                                },
                                "folder": {
                                    "type": "array",
                                    "title": "Folders",
                                    "description": "Hierarchical folders
to identify the fact",
                                    "items": {
                                        "type": "string",
                                        "title": "Folder",
                                        "description": "Fact Folder to
use for the fact selection"
                                    }
                                }
                            },
                            "required": [
                                "id",
                                "folder"
                            ]
                        }
                    },
                    "objectTypes": {
                        "type": "array",
                        "title": "Object Type Selections",
                        "description": "Definition of a Object Type
Selection",
                        "items": {
                            "type": "object",
                            "properties": {
                                "id": {
```

```
                                    "type": "string",
                                    "title": "Object Type Identifier",
                                    "description": "Object Type
Identifier to use for the selection"
                                },
                                "version": {
                                    "type": "string",
                                    "title": "Version",
                                    "description": "Version of the
Object Type to use for the selection"
                                }
                            },
                            "required": [
                                "type",
                                "version"
                            ]
                        }
                    }
                }
            },
            "required": [
                "domain",
                "package",
                "dimensions",
                "facts"
            ]
        }
    },
    "required": [
        "packagesConfiguration"
    ]
}
```

6.36 Unified Data Response

The response received from the UOC plugin has the following definition in JSON. The response will contains values over the time if the query integrate a granularity parameters (e.g. values every 15 minutes). If the granularity is part of the query, then the response will integrate a column to manage the associated timestamp and additional option to identify the exact data that match this timestamp. **<install_dir>/json-schema/unified-data-response.json**

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
```

```
    "type": "object",
    "title": "Unified Data Response",
    "description": "Definition of a data response from the UOC plugin.
Note if the granularity is part of the query, then it will require a
specific timeColumn property to identify the column in charge of the
timestamp",
    "properties": {
        "headers": {
            "type": "array",
            "title": "Headers",
            "description": "Specification of the returned columns. It
defines dimensions or fact identifiers, and timestamp if a granularity
is requested in the query",
            "items": {
                "type": "string",
                "title": "Data Definition",
                "description": "Identifier or dimension, fact or
timestamp column"
            }
        },
        "values": {
            "type": "array",
            "title": "Values",
            "description": "Response of the query following the order of
the header definition. Type can be string, integer, boolean",
            "items": {}
        },
        "status": {
            "type": "string",
            "title": "Status",
            "description": "Return string of the query. It could be OK
or KO"
        },
        "nbVals": {
            "type": "integer",
            "title": "Number of values",
            "description": "Number of values returned in the response.
It is the page size if the query is server side paginated"
        },
        "totalCount": {
            "type": "integer",
            "title": "Total Count",
            "description": "Total number of values for the query."
        },
```

```
        "timeColumn": {
            "type": "string",
            "title": "Time Column Identifier",
            "description": "Only mandatory if the query integrated a
granularity request. It specified the column of the header in charge of
the timestamp"
        }
    },
    "required": ["headers", "values", "status", "nbVals", "totalCount" ]
}
```

6.37 Action Navigation

The object action navigation has the following definition in JSON. <install_dir>/json-schema/action-navigation.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Action Navigation",
    "description": "Definition of the navigation action (by url or by
View Identifier)",
    "properties": {
        "id": {
            "type": "string",
            "title": "Identifier",
            "description": "Unique identifier of the action"
        },
        "name": {
            "type": "string",
            "title": "Name",
            "description": "Presentation Name of the action"
        },
        "type": {
            "type": "string",
            "title": "Type",
            "description": "Type of the action. For this action, it is
navigation",
            "default": "navigation"
        },
        "oneOf": [{
            "view": {
                "type": "string",
```

```
                "title": "View",
                "description": "View Identifier to navigate to"
            }
        }, {
            "url": {
                "type": "string",
                "title": "Url",
                "description": "URL to navigate to"
            }
        }]
    },
    "required": [ "id", "name", "type" ]
}
```

6.38 Action Parameter

The object action parameter has the following definition in JSON. <install_dir>/json-schema/action-parameter.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Action Parameter",
    "description": "Definition of the action parameter to define output
parameters in the Data Exchange Service",
    "properties": {
        "id": {
            "type": "string",
            "title": "Identifier",
            "description": "Unique identifier of the action"
        },
        "name": {
            "type": "string",
            "title": "Name",
            "description": "Presentation Name of the action"
        },
        "type": {
            "type": "string",
            "title": "Type",
            "description": "Type of the action. For this action, it is
parameter",
```

```
            "default": "parameter"
        },
        "$ref": "#output-parameters"
    },
    "required": [ "id", "name", "type" ]
}
```

6.39 Action Object Type

The object action objectType has the following definition in JSON. <install_dir>/json-schema/action-object-type.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Action Object Type",
    "description": "Definition of the action objectType to popup a form
to perform a synchronous or asynchronous operation",
    "properties": {
        "id": {
            "type": "string",
            "title": "Identifier",
            "description": "Unique identifier of the action"
        },
        "name": {
            "type": "string",
            "title": "Name",
            "description": "Presentation Name of the action"
        },
        "type": {
            "type": "string",
            "title": "Type",
            "description": "Type of the action. For this action, it is
objectType",
            "default": "objectType"
        },
        "executionType": {
            "type": "string",
            "title": "Type of execution for the operation",
            "description": "Type of execution of the operation:
synchronous or asynchronous",
            "default": "asynchronous"
        },
```

```
        "$ref": "#data-selection"
    },
    "required": [ "id", "name", "type", "dataSelection" ]
}
```

6.40 Action Launch

The object action launch has the following definition in JSON. <install_dir>/json-schema/action-launch.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Action Launch",
    "description": "Definition of the action launch to popup an external
application",
    "properties": {
        "id": {
            "type": "string",
            "title": "Identifier",
            "description": "Unique identifier of the action"
        },
        "name": {
            "type": "string",
            "title": "Name",
            "description": "Presentation Name of the action"
        },
        "type": {
            "type": "string",
            "title": "Type",
            "description": "Type of the action. For this action, it is
launch",
            "default": "launch"
        },
        "launch": {
            "type": "string",
            "title": "Launch Identifier",
            "description": "Identifier of the launch to execute"
        }
    },
    "required": [ "id", "name", "type" ]
}
```

6.41 Action Module

The object action module has the following definition in JSON. <install_dir>/json-schema/action-module.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Action Module",
    "description": "Definition of the action module to call a custom
module",
    "properties": {
        "id": {
            "type": "string",
            "title": "Identifier",
            "description": "Unique identifier of the action"
        },
        "name": {
            "type": "string",
            "title": "Name",
            "description": "Presentation Name of the action"
        },
        "type": {
            "type": "string",
            "title": "Type",
            "description": "Type of the action. For this action, it is
module",
            "default": "module"
        },
        "module": {
            "type": "string",
            "title": "Module Identifier",
            "description": "Identifier of the module to call"
        },
        "anyOf": [{
            "description": "Define here your own properties that can
only be managed by your module"
        }]
    },
    "required": [ "id", "name", "type" ]
}
```

6.42 Action Command

The object action command has the following definition in JSON. <install_dir>/json-schema/action- command.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Action Command",
    "description": "Definition of the action Command to execute an
internal UOC command",
    "properties": {
        "id": {
            "type": "string",
            "title": "Identifier",
            "description": "Unique identifier of the action"
        },
        "name": {
            "type": "string",
            "title": "Name",
            "description": "Presentation Name of the action"
        },
        "type": {
            "type": "string",
            "title": "Type",
            "description": "Type of the action. For this action, it is
command",
            "default": "command"
        },
        "commandId": {
            "type": "string",
            "title": "Command Identifier",
            "description": "Identifier of the command to execute"
        }
    },
    "required": [ "id", "name", "type" ]
}
```

Chapter 7
Packages (Value Packs)

Packages is a key concept for UOC and data integration. The packages represents the business added values in term expertise on a specific domains. A package is easy to install, modify and resell in solutions.

UOC is a value-pack driven application that can be easily extended with new information, graphical screens, etc. installing new value-packs.

7.1 Overview

The packages (or value packs) is an unified format that expose to the Unified Console, the definition of data for a given domain (i.e. a dedicated plugin). UOC uses this value packs to identify which plugin is able to retrieve the requested data from client side components (i.e. widgets, modules, menus)

A Value Pack consists of :

- One or more workspaces, with a set of preconfigured views (pages) i.e. pre-defined graphical screens for UOC integration.
- The model of the data (meta data) exposed by the domain server:
  - Dimensions / facts definitions and their relationships.
  - Objects Definition / Operations / Forms

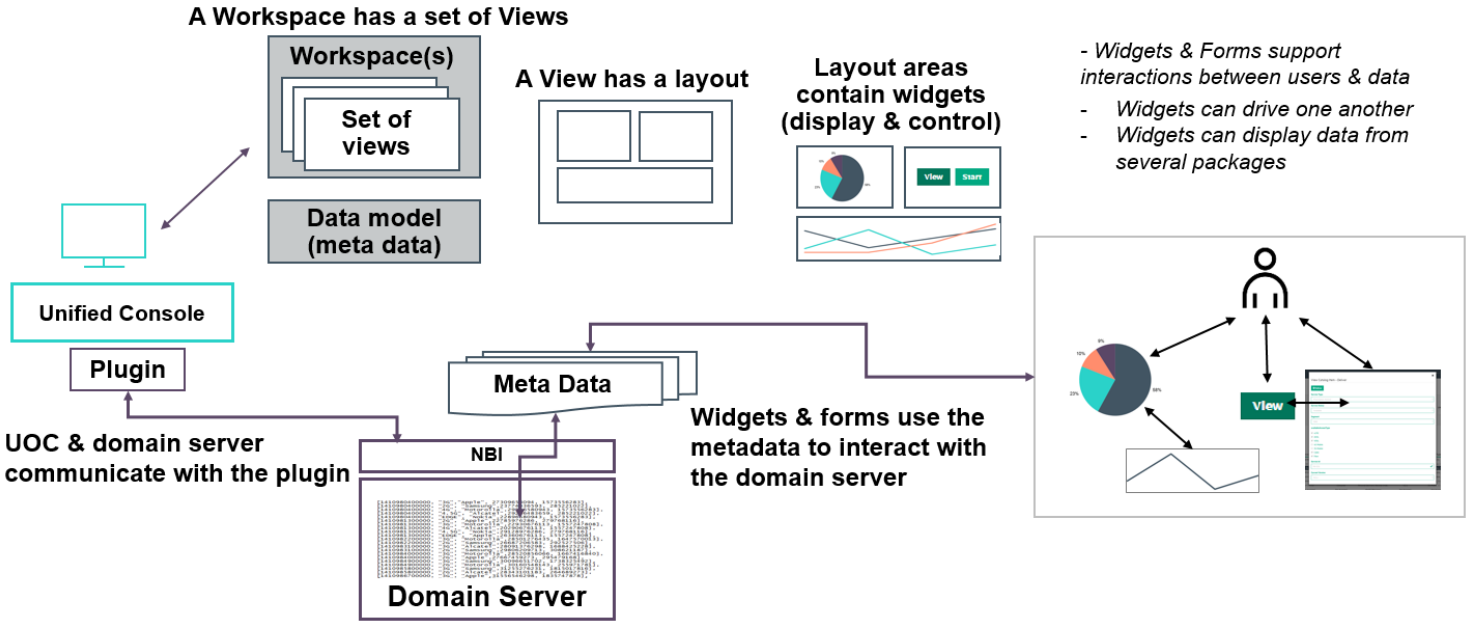


Figure 59 – Packages Overview

Example of value pack:

- Mobile Data 3G & LTE
- Voice & messaging 2G, 3G
- VOLTE
- Fault Management, etc…

7.2 Structure

Its basic structure is composed of 3 parts:

- Metadata. It is internal definition never exposed at the UOC level. It is usually linked to a data table, or aggregation rules provided by the data server.
- Metadata for user interface (package) (It needs to match a data server). It defines what will be exposed at the user interface in terms of dimensions and facts (metrics).
- Data for the user interface UOC (views and/or workspaces) (optional). It is composed of pre-defined screens for operators. These screens can be customized later. It is a way to provide some templates corresponding to a value pack's content, to speed up the deployment of the value pack inside UOC.

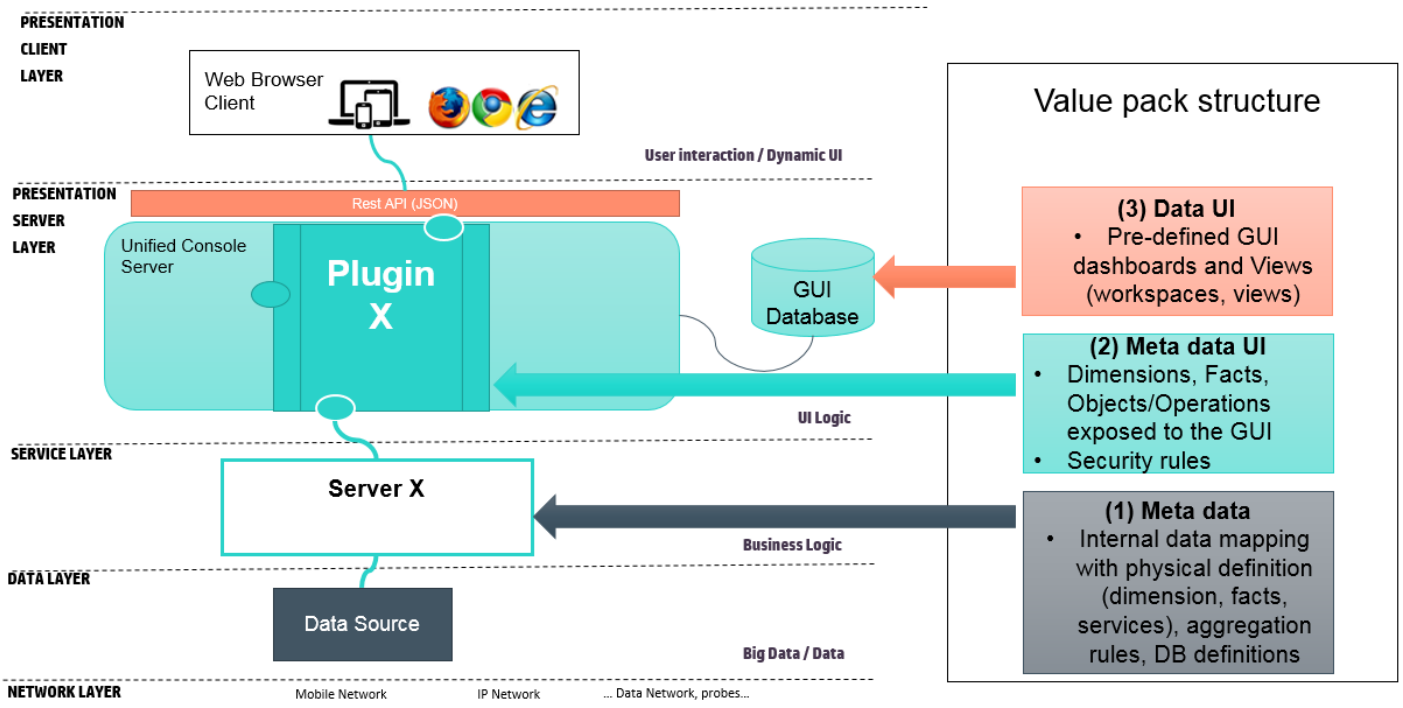


Figure 60 – Value pack structure

Here is an example of a value pack used to monitor customers through graphical dashboards and to manage subscriptions to services. Objects can be defined to support CRUD operations and dimensions and metrics (facts) can be defined accordingly to display summarized information.

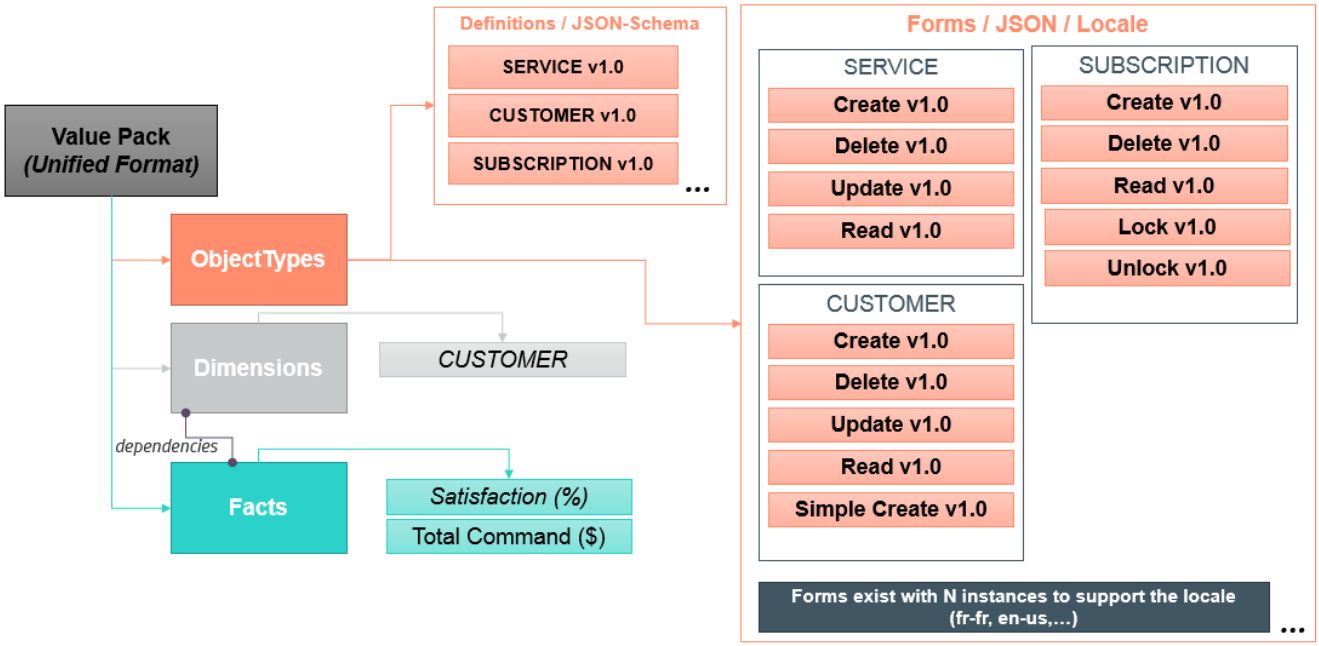


Figure 61 – Example of Value pack to monitor customers and subscriptions

7.3 Object Types and Operations format

This format needs an object type and its supported operations definition and form definitions

It is possible to define multiple objects in the value pack (identifier, name and version) and associate security rules to them to access to this information (RBAC). The definition of this object is based on an external JSON-Schema definition where all the attributes are defined as well as available operations on these objects.

All objects can have one or many GUI form definitions to indicate to the UOC components how to display or request information.

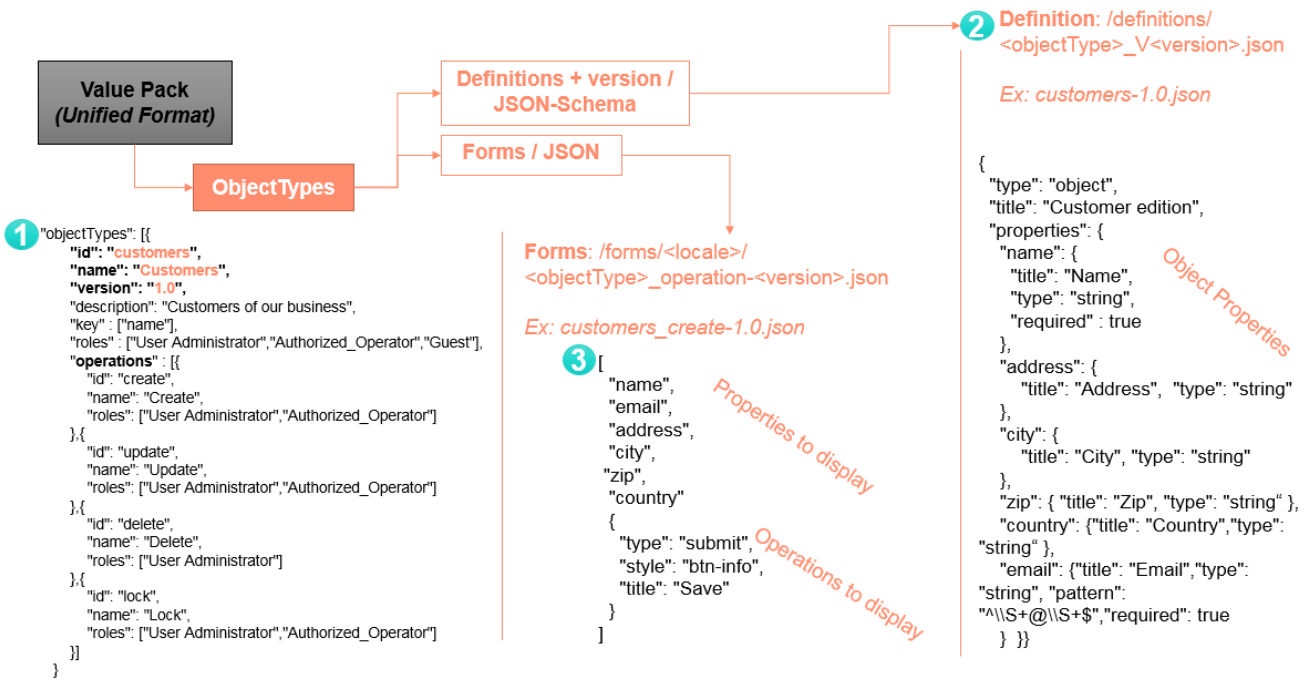


Figure 62 – Example of Object and operation support in value pack

7.3.1 Object and Operations definition

The package describes all the objects available in this available and their associated operations. These operations can be protected by roles.

Example

```
"objectTypes": [{
        "id": "customers",
        "name": "Customers",
        "version": "1.0",
       "description": "Customers of our business",
       "key" : ["name"],
       "roles" : ["User Administrator","Authorized_Operator","Guest"],
       "operations" : [{
          "id": "create",
          "name": "Create",
          "roles": ["User Administrator","Authorized_Operator"]
       },{
          "id": "update",
          "name": "Update",
          "roles": ["User Administrator","Authorized_Operator"]
       },{
          "id": "delete",
          "name": "Delete",
          "roles": ["User Administrator"]
       },{
          "id": "lock",
          "name": "Lock",
          "roles": ["User Administrator","Authorized_Operator"]
       }]
   }
```

7.3.2 Object Definition

A dedicated JSON file describe each objects and their properties (name, required, type, etc.)

Example

```
{
  "type": "object",
  "title": "Customer edition",
  "properties": {
    "name": {
      "title": "Name",
      "type": "string",
      "required" : true
    },
    "address": {
        "title": "Address",  "type": "string"
    },
    "city": {
        "title": "City", "type": "string"
    },
    "zip": { "title": "Zip", "type": "string“ },
    "country": {"title": "Country","type": "string“ },
    "email": {"title": "Email","type": "string", "pattern": "^\\S+@\\S+$","required": true}
}}
```

7.3.3 Form

To perform operations using form, a dedicated JSON file is required to describe how the objects and its properties will be displayed to the end user.

Example

```
[
 "name",
 "email",
 "address",
 "city",
"zip",
 "country"
 {
   "type": "submit",
   "style": "btn-info",
   "title": "Save"
 }
]
```

7.4 Dimensions / Facts format

Several dimensions can be defined in the value pack for analysis, and several facts can be detailed for these dimensions (type, unit, ordering…) for dashboard usage.

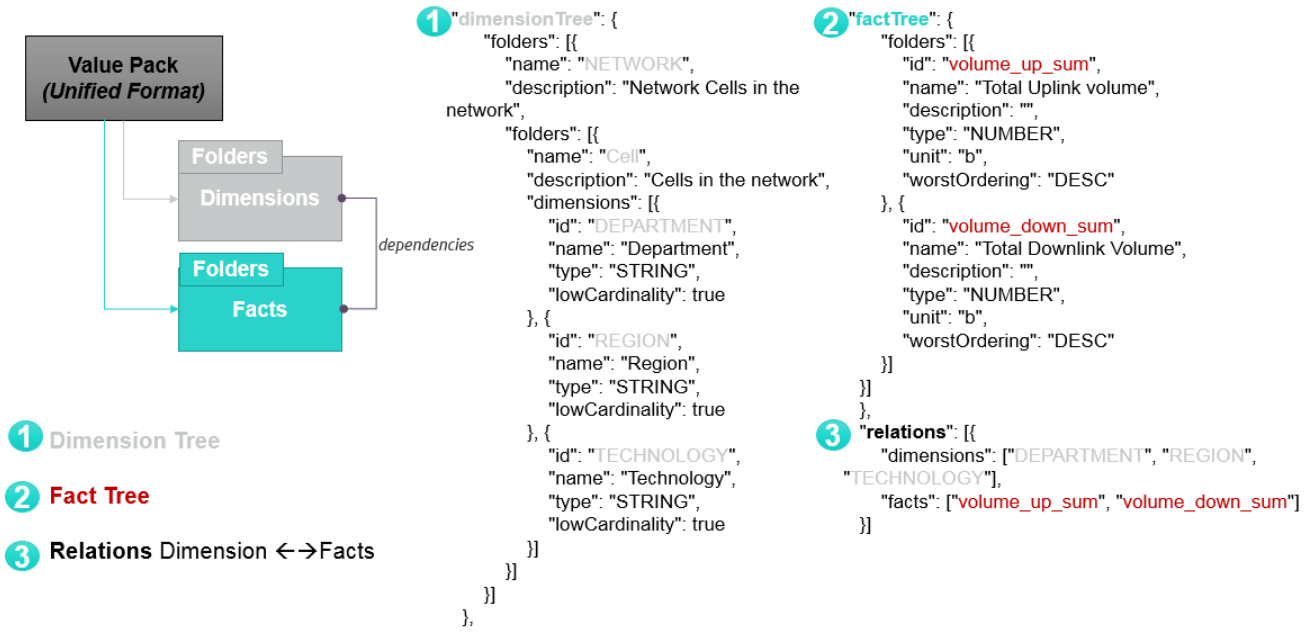
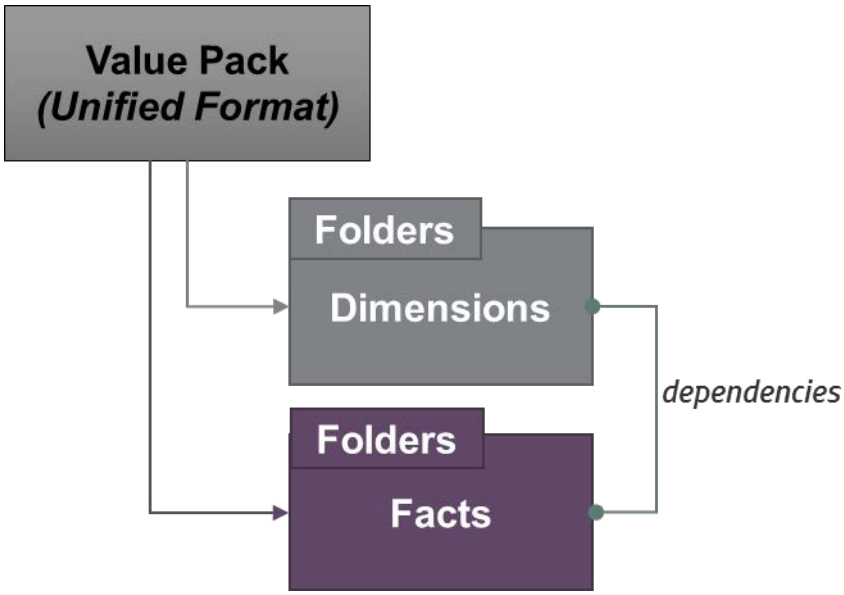


Figure 63 – Example of Dimensions/Facts support in value pack

7.4.1 Description



This format needs definitions for:

1. Dimension Tree
2. Fact Tree
3. Relations between Dimension and Facts

7.4.2 Dimension Tree

The Dimension tree is organized by folders (and sub folders) and describes the list of available dimension (i.e. criteria of analysis to get metrics)

Example

```
"dimensionTree": {
    "folders": [{
      "name": "NETWORK",
      "description": "Network Cells in the network",
      "folders": [{
        "name": "Cell",
        "description": "Cells in the network",
        "dimensions": [{
          "id": "DEPARTMENT",
          "name": "Department",
          "type": "STRING",
          "lowCardinality": true
        }, {
          "id": "REGION",
          "name": "Region",
          "type": "STRING",
          "lowCardinality": true
        }, {
          "id": "TECHNOLOGY",
          "name": "Technology",
          "type": "STRING",
          "lowCardinality": true
        }]
      }]
    }]
  },
```

7.4.3 Fact Tree

The filter tree is organized by folder and described all the metrics associated to dimensions.

Example

```
"factTree": {
    "folders": [{
      "id": "volume_up_sum",
      "name": "Total Uplink volume",
      "description": "",
      "type": "NUMBER",
      "unit": "b",
      "worstOrdering": "DESC"
    }, {
      "id": "volume_down_sum",
      "name": "Total Downlink Volume",
      "description": "",
      "type": "NUMBER",
      "unit": "b",
      "worstOrdering": "DESC"
    }]
  }]
  },
```

7.4.4 Relations

This part describe the association between dimension (from the dimension tree) and facts (from the fact tree). This is useful to provide an intelligent analysis tool that will show and hide dimensions and facts in a very consistent way, and then avoid invalid query to data servers.

Example

```
"relations": [{
    "dimensions": ["DEPARTMENT", "REGION", "TECHNOLOGY"],
    "facts": ["volume_up_sum", "volume_down_sum"]
  }]
```

7.5 Notifications

Packages can define notifications and integrate them in a dedicated JSON file.

The detailed description can be found in 6.12 Notifications

```
    "notifications": [{
        "type": "alert",
        "id": "operation_readiness-already_started",
        "roles": ["Administrator", "View Designer", "Platform
Administrator", "User Administrator", "Package Designer",
"Delivery_Manager"],
        "title": "Service already started",
        "update": false,
        "level": "warn",
```

```
        "actions": [],
        "keywords": ["Healthcare", "Operation Readiness"]
    }
… ]
```

7.6 Workspaces

Packages can pre-defined workspaces to associate to a set of metadata. These workspaces will be automatically added in the GUI database at startup or after a reload package operation (dynamic refresh).

These workspaces can be built sing the GUI Workspace Manager or manually editing JSON. Please refer to the detailed description in 6.7 Workspaces.

You can easily browse associated workspace to packages using the Package Manager and click on your package name.

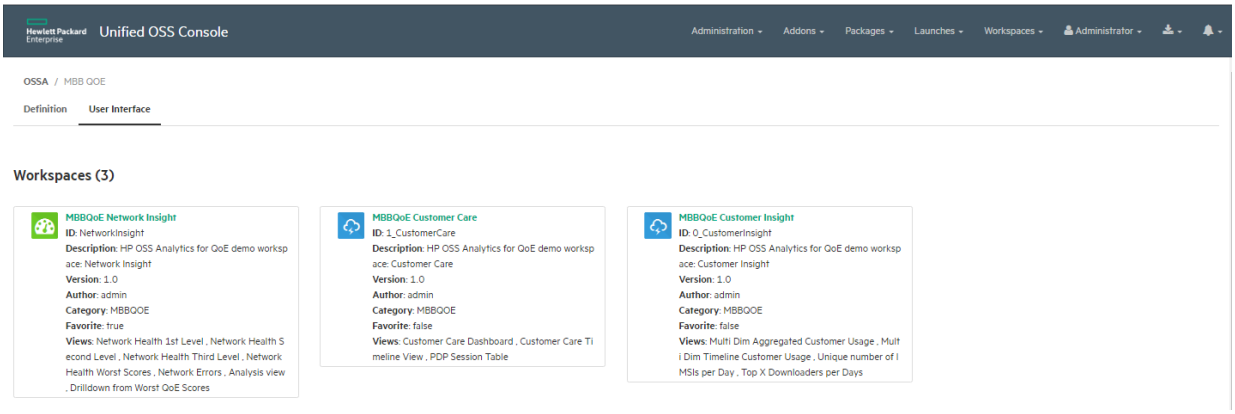


Figure 64: Package Manager - User Interface – Workpaces

7.7 Views

Packages can pre-defined views to associate to a set of metadata. These views will be automatically added in the GUI database at startup or after a reload package operation (dynamic refresh).

These views can be built using the View Designer or manually editing JSON. Please refer to the detailed description in 6.8 Views

You can easily browse associated views to packages using the Package Manager and click on your package name.

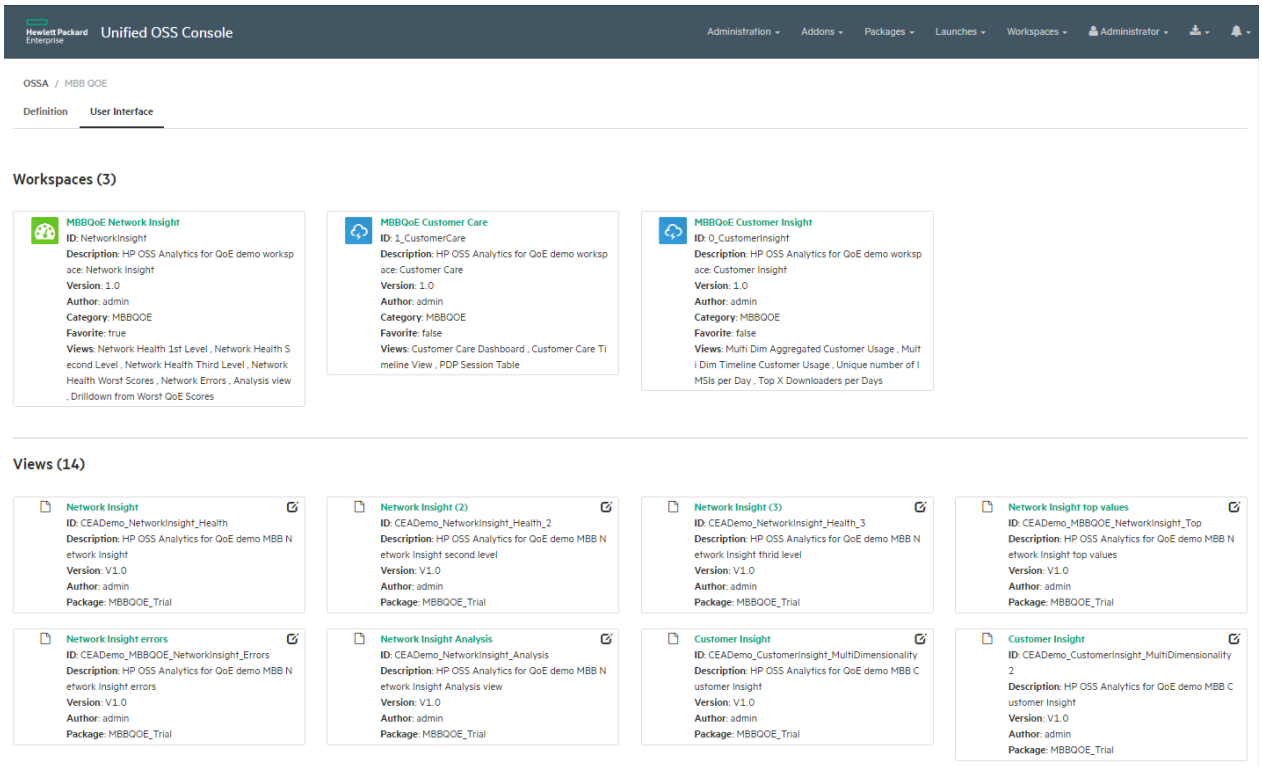


Figure 65: Package Manager - User Interface - Views

7.8 Value Pack Management

During the startup of the UOC server, through activated plugins exposing their associated value packs, the UOC server gets dynamically all the value packs available on data servers, identifies what kind of data is available on these specific servers and extends dynamically the user interface to support these value packs at the end user level. During the import:

- The metadata (1) are ignored and stay on the data server. This cannot be accessed by UOC application's client-side.
- The metadata UI (2) also called "package" in the UOC application are dynamically loaded and stored in memory of UOC server (in its associated plugin).
- The Data UI (3) basically composed of pre-defined graphical views, and pre-defined workspaces ready to be open, are imported (or updated) in the GUI document database. The UOC application will always use the database (Apache CouchDB) to find its available graphical workspaces and views.

Example with the OSS Analytics plugin.

The Mobile Broad Band QoE value pack (MBBQoE) is loaded into the UOC Server and ready for use, and all associated workspaces and views prepared by a View Designer / Package Designer are imported (or updated) into the GUI document database (Apache CouchDB). After this startup phase, the MBBQoE value pack is available. The operator can open the new workspaces, display new views, and is ready to use the analysis tools to troubleshoot quality of service for the mobile domain.

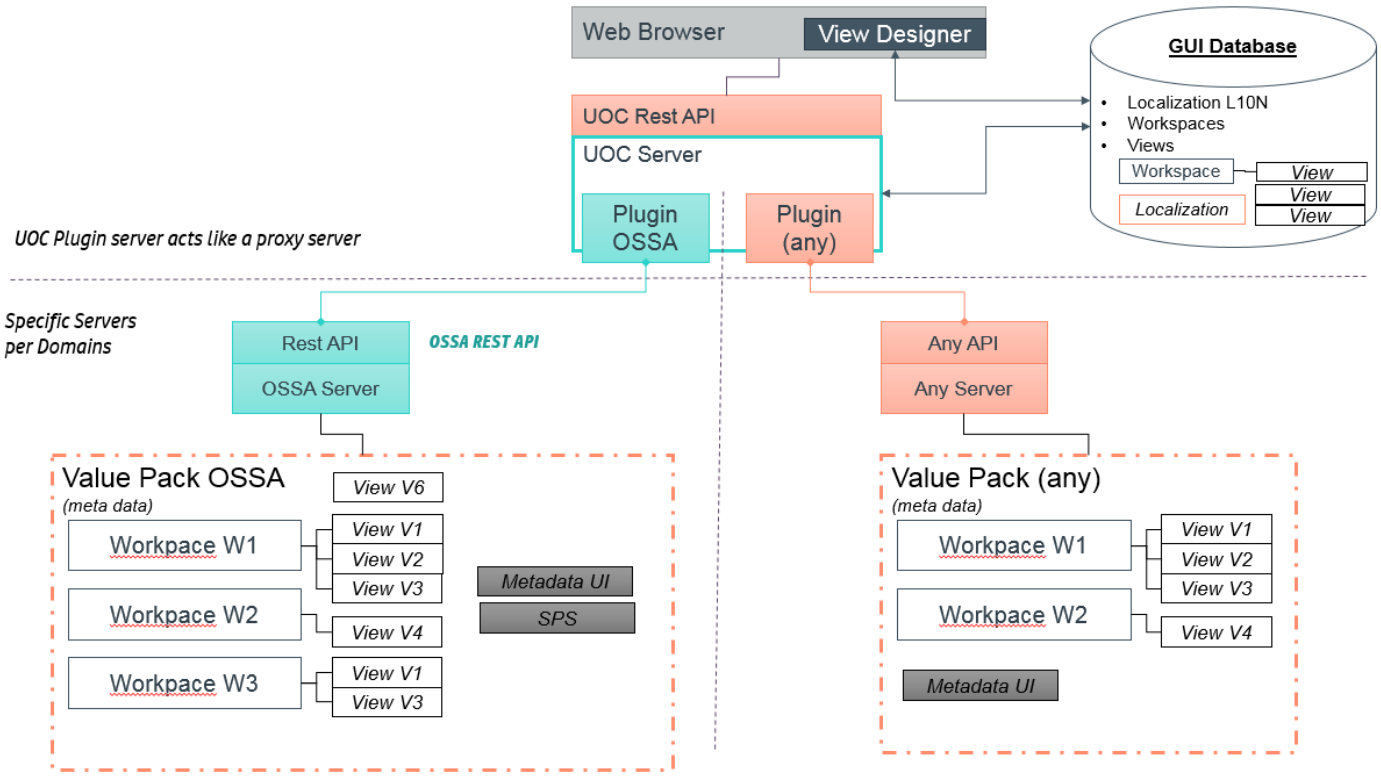


Figure 66 – Value Pack Management

The definition of these objects are requested from the plugin (official RESTAPI) in charge of the value pack (JSON-Schema) with an identifier and version.

NOTE: A startup import policy is available for customization (see the Installation Guide for details). By default, all the packages received from data servers are never overwritten by local data.

Chapter 8
Server Add-ons – Plugin

This chapter details the purpose of the plugin and technical details. This is a key component in a solution based on the Unified Console to access information.

To start implementing a new plugin, it is recommended to use the SDK generator to generate a working skeleton (see 5.9 How to generate an add-on server Plugin), and then customize this new plugin to fit your own needs.

You also can use a special plugin provided by the SDK named plugin simulator to bootstrap a development until you can access real data and data servers. This plugin simulator is an ideal component to build a demo, proof of concept, etc.

8.1 Overview

A plugin is a key component of UOC's architecture. It acts as an adaptor between the UOC application and one or multiple data servers for a specific domain (Customer Experience Assurance, Fault Management, OSS Analytics, NFV Director, NFV Analytics…).

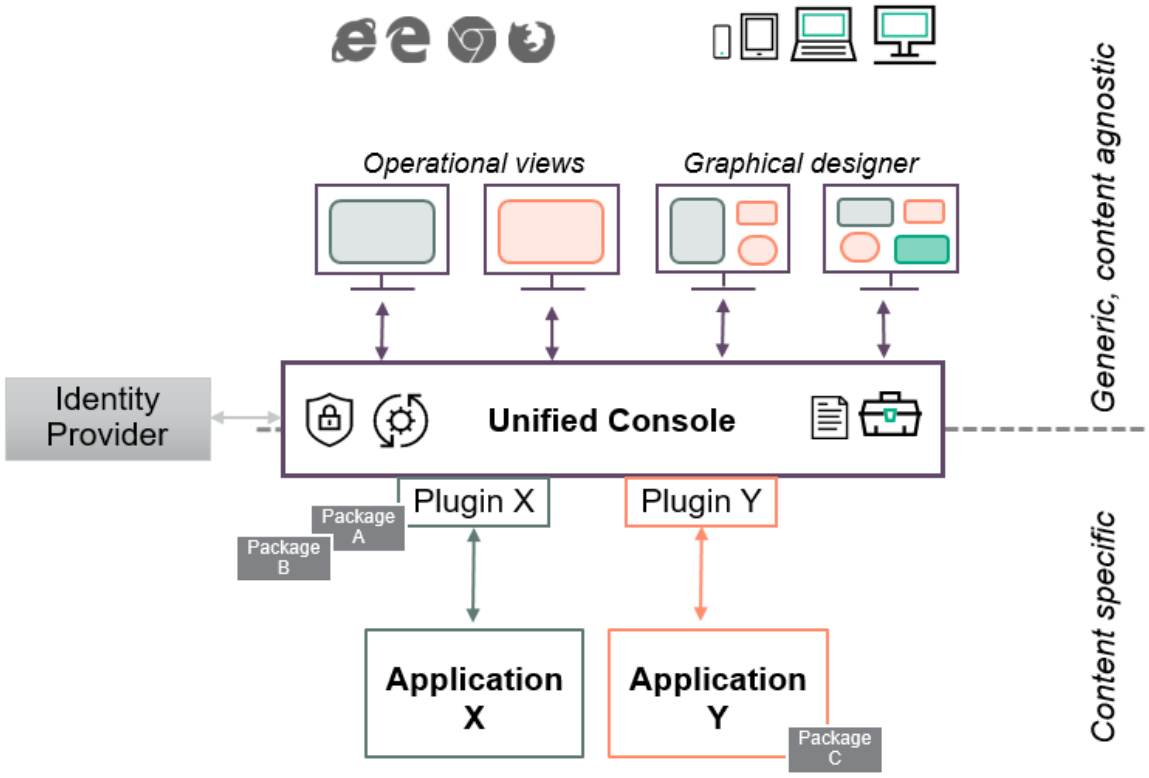


Figure 67 – Plugin Overview

This add-on server component is in charge of communicating with the server i.e. querying information, querying value packs (or packages), views, workspaces… and executing operations on the data server (updating a state, creating an object, etc.).

A plugin is specific to a data server (or multiple data servers using the same interface) and have to implement the right interfaces to retrieve all the needed information. These interfaces can be heterogeneous, custom, etc… The only constraint of the plugin is to expose a unified format to the UOC server.

This plugin component is an interface to ease dialog with a data server for the UOC application. All widgets will request information through the plugin of UOC server and never call directly a data server.

It allows the architecture to be more flexible and extremely scalable.

Each plugin has the following purposes:

- Key component is in charge of the content specific implementation based on the northbound interface exposed by one or multiple data servers it want to connect to. This northbound interface could be any protocol or custom API. The plugin is the only component in the solution that has a specific knowledge of the interfaces of the data servers. It hides the complexity of the servers accesses in the solution. There are several use cases to plug your own data server(s) and data source(s) into Unified Console:
  - one data server (one data source)
  - multiple data servers (multiple data sources)
  - aggregate information from multiple data servers (aggregated data sources to simplify the exchange between UOC and the data servers)
- Implement a mandatory interface (REST API) to grant that UOC can access to any of these plugins. From UOC point of view, leveraging such common interface provide a way to have a generic and content agnostic coupling between the client add-on components (widgets, modules…) and its associated servers in charge of sending needed information. See 8.5 Rest API.
- Implement the conversion of formats between the data servers and UOC to fit the Unified Format. See 6.36 Unified Data Response.
- Manage the Security and Access Control to send these information to the data server in case you need to implement their own data obfuscation.
- Define one or multiples packages (or value packs) definition that will details all the needed information for UOC as logical data model and providing to UOC an abstract layer. UOC is value-pack driven for all its queries. Each package presents data definition (metadata) and associated GUI (pre-defined views and workspaces ready to use).

  Storage of packages is very flexible and depends on your own need:

  - A package can be manage by the plugin itself as static JSON file definition (metadata) in **<install_dir>\server\public\addons\<my plugin Id>\packages\<my package id>.** It is an easy solution.
  - A packages can be requested from a data server and send to UOC as JSON file definition (metadata). Your package reference is closed to your data on the same data server.
  - A package can be dynamically generated to produce at runtime the correct metadata definition to UOC. This could be useful if a reference of data already exist in a different format or if the data t is very dynamic.

NOTE: Only the Plugin REST API is called from UOC. So the REST API implementation to manage packages, views, workspaces, objects definition and forms are very flexible and highly customizable.

The configuration information needed for the plugin is stored in its associated configuration file (JSON) in **<install_data_dir>/server/public/addons/plugins/<pluginId>/config.json**

Usually, the plugin can be enabled or disable by configuration. An “active” flag in the configuration of the plugin can be set to True or False. If the Active flag is False, the plugin is not loaded during UOC Server startup and not available for dashboards or views.

Some plugins, like OSS Analytics plugin, have advanced configuration because they manage several data server configurations sharing the same interface. So the configuration definition provides multiple servers, protocols, ports definition.

NOTE: HTTP compression is enabled by default between the plugin and UOC.

It is recommended to check that this compression is also enabled between your plugin and your data server to increase significantly the performance.

8.2 How to use a plugin ?

8.2.1 Plugin managing a single data server

in this use case, the plugin is in charge of only on data server i.e. one data source. Integrator needs to define properly the package associate to this data server and manage it through the common REST API at plugin level.

The plugin uses the northbound interface of the data server and implement queries, response and mandatory REST API of the plugin.

Any protocol can be supported be plugin and its data server. It may require to integrate 3rd party products or open sources to ease the development.

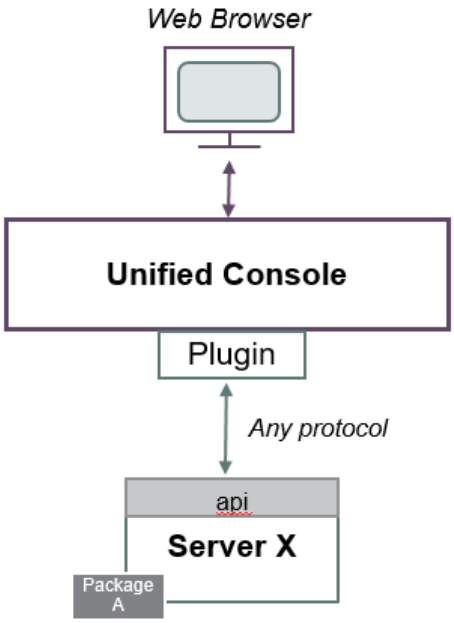


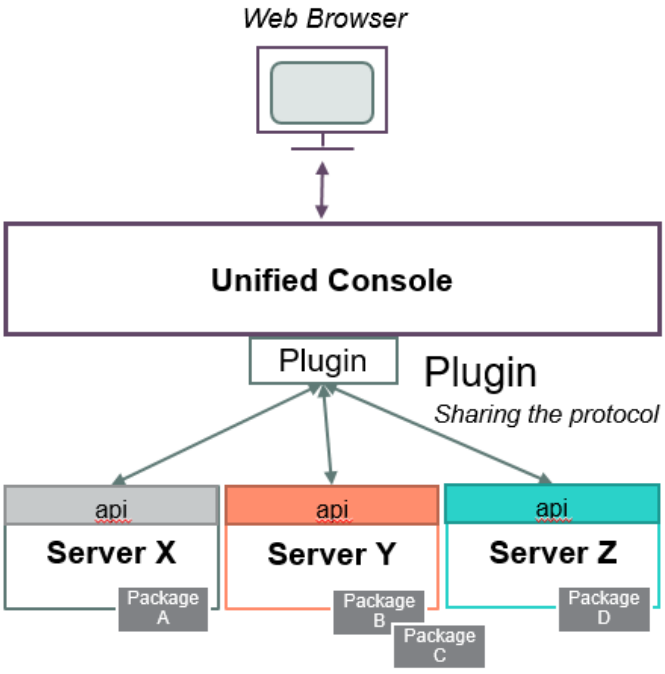
Figure 68 – Plugin managing single date server

8.2.2 Plugin managing multiple data servers

In this use case, the plugin is in charge of providing an interface to several data servers i.e. multiple data sources. There is no dependencies on these data sources, and each servers will expose its packages to indicate to UOC what kind of information each of them is able to provide.

The plugin uses the northbound interface of the data servers and implement queries, response and mandatory REST API of the plugin. This interface is the same for all these data servers, and the package definition helps UOC to target the right data servers for queries. Usually, these packages are split to provide different information and UOC will be in charge to correlate these data on the GUI.

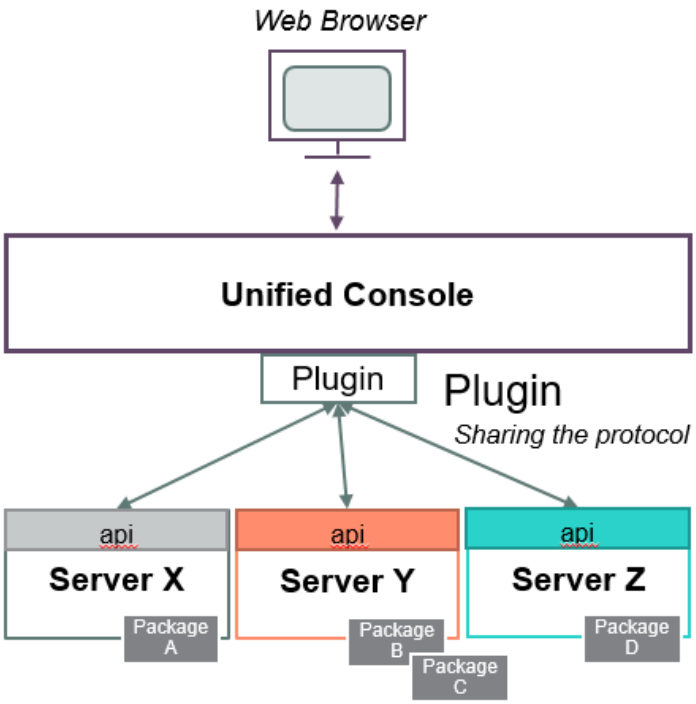
Any protocol can be supported by the plugin and its data server but it must be the same between all these servers.. It may require to integrate 3rd party products or open sources to ease the development.



It is typically the use case for OSS Analytics (OSSA). The plugin OSSA support in its configuration multiple data server configuration. Based on the package and data requested, queries will be sent to the right data servers and provided to UOC for display.

Figure 69 – Plugin managing multiple date servers

8.2.3 Plugin managing aggregated data servers



In this use case, the plugin is in charge of providing an interface to several data servers i.e. multiple data sources but from UOC point of view, it will be like having only one data source. Response provided to UOC will be aggregated between response of several data servers. It is usually the case when there is a dependencies between information sent to UOC. Goal is to hide complexity of these dependencies and expose 'one' source to UOC i.e. one package with aggregated information.

The plugin uses the northbound interface of the data servers and implement queries, response and mandatory REST API of the plugin. This interface can be different between these data servers and it will require a configuration able to define all these options.

Any protocol can be supported by the plugin and its data server and it can different one between all these servers..
It may require to integrate 3rd party products or open sources to ease the development.

Figure 70 – Plugin managing multiple date servers

Example:

The package expose an complex object composed on several part (status, fulfillment information) . Information for these parts are split between several systems and required different interface to access to these data. The plugin will simplify the interaction between the UOC and these data servers. As soon as the plugin identify which data servers need to be queried, it can execute several request asynchronously to these servers, wait for responses, and when all responses are received, build a aggregated response to UOC.

From UOC point of view, it simy queries information on an object using its identifier. Package definition is simple as an abstract layer and UOC does not have to take care of dependencies to build its GUI screens. The complexity and physical view of information stay at plugin level.

It is typically the use case for NFV Director, Service Director, etc. that requires several servers in charge of fulfilment, activation, monitoring, etc.

8.3 Structure

Structure of a plugin is very flexible as soon as you implement the required RESTAPI (see 8.5 Rest API) but usually the standard structure of a plugin is composed of several parts:

- **Implementation of the plugin** that will be place under <install_dir>\server\addons\<my plugin Id> directory and will be composed of the following files:

Files	Description
plugin.json	Descriptor of the plugin. It contains all the information to identify the plugin and its general information (version, author, description, etc.)
constants.js	Definition of the constant used in this plugin
routes.js	Definition of all URL supported by this plugin. These URL must follow the unified format requested by a plugin (see 8.5 Rest API)
package-manager.js	Implementation of the routes (URL) to provide request yoru data server, format response and send unified information to UOC.
package.json	NPM's package definition to define external dependencies and versions. By default is empty, but you may have to add new third party product as external dependencies. See 8.9 Integrate an external Node Module in your plugin
README.txt	Simple read me file to add details about your plugin.

- **Configuration files** of the plugin that will be place under <install_dir>\server\public\addons\<my plugin Id> directory and will be composed of the following files:

Files	Description
config.json	Configuration file used during development. This format is flexible and should be adjusted based on your need. It usually contains all the information mandatory to initiate a connection to your data server(s)
config.json.kitting	Configuration file used for packaging and kitting. Default config.json you want to install by default.

- **Packages definition** of the plugin that will be place under <install_dir>\server\public\addons\<my plugin Id>\packages\<package Id> directory and will be composed of the following files:

Files	Description
Metadata	Directory to store metadata information
• package.json	Definition of the value pack i.e. all the data exposed to UOC by this plugin
gui	Directory to store graphical object of the package
• views	Directory to store all the definition of views defined for this package
• workspaces	Directory to store all the definition of the workspaces for this package

The generator provided in the SDK to generate plugin and/or package will follow this structure. See the following section for detailed information:

- 5.8 How to generate an add-on server Package
- 5.9 How to generate an add-on server Plugin

IMPORTANT: The code of the plugin can be replaced by a patch or upgrade, but files store in <install_dir>/data, <install_dir>/client/public or <install_dir>/server/public are never uninstall or replaced.

This grant that after an upgrade of the product, there is not lost of configuration, or customization.

8.4 Descriptor

The descriptor defines the plugin object. Please refer to the object JSON definition in 6.13 Plugins

Example of OSS Analytics Descriptor

```
{
   "id": "ossa",
   "name": "OSSA",
   "version": "1.1",
   "description": "Plugin for OSS Analytics Server",
   "author": "HPE"
}
```

8.5 Rest API

The Plugin REST API is the only API used between UOC and the plugin. There are several type of RESTAPIs:

- Common REST API: Mandatory RESTAPI dedicated to UOC to be able to load plugin descriptor, packages, views, workspaces, etc.
- Dimension / Fact REST API: to manage the Dimension / fact format very useful for analytics use cases
- Objects / Operations REST API: to manage ascending and descending operations on objects (CRUD)
- Self-Monitoring RESTAPI: dedicated to ease the platform monitoring and detect quickly which component may have issues in production.
- Packages REST API: Advanced operation on packages like dynamic reload of a given packages, or all packages of the plugin.

8.5.1 Common REST API

These URLs manage the common information related to the plugin, packages, workspaces and views.

Description	HTTP Verb	Rest API
Get the plugin descriptor	GET	/V1.0/domains/<pluginId>
Get the definitions of available packages	GET	/V1.0/domains/*<domainId>*/packages
Get the package definition for a specific package	GET	/V1.0/domains/*<domainId>*/packages/<packageId>
Get all workspaces definition for a specific package	GET	/V1.0/domains/*<domainId>*/packages/*<packageId>*/workspaces
Get all views definition for a specific package	GET	/V1.0//domains/*<domainId>*/packages/*<packageId>*/views
Get all views definition for all packages	GET	/V1.0/packages/views
Get all workspaces definition for all packages	GET	/V1.0/packages/workspaces
Get a specific package definition	GET	/V1.0/packages/*<packageId>*

8.5.2 Dimensions / Facts REST API

These URLs manage the dimensions and facts model to retrieve values of facts.

Description	HTTP Verb	Rest API
Return all possible distinct values of a dimension or a combinaison of dimension	GET	/V1.0/packages/*<packageId>*/dims/*<dim1,dim2…dimN>*/distinct
Return all possible values of a dimension or combination of dimensions matching given expression patterns *string*string*)	GET	/V1.0/packages/*<packageId>*/dims/*<dim1,dim2…dimN>*/match/*<expression>*
Return data query based on a list of dimensions and facts	GET	/V1.0/packages/*<packageId>*/facts/*<fact1,fact2…factiN>*/dims/*<dim1,dim2…dimN>*
Return data query based on a list of facts	GET	/V1.0/packages/*<packageId>*/facts/*<fact1,fact2…factN>*

8.5.2.1 Unified Format (with granularity)

Format returned by the plugin depends if the granularity (i.e. date/time period) is requested or not. If the request needs to see a value over the time, it will perform a request with a granularity (like 1h, 15 min, 24h…). So, granularity means you will get the historical values every N seconds and a timestamp is present in each items of the response.

TIP: Ideal for all graphical representation that displays values over the time like line, spline, bar charts…

Example:

http://localhost:9000/V1.0/domains/ossa/packages/MBBQOE_Trial/facts/failed_DNS_req_count/dims/TECHNOLOGY?begin=1418688000000&end=1426546799000&granularity=900&batchsize=100000

will return a column for all dimensions (Technology) , facts (Failed DNS req count) and the time (event timestamp)


```
{
   "headers": ["EVENT_TIMESTAMP", "TECHNOLOGY", "failed_DNS_req_count"],
   "values": [
      [1424074500000, "3G", 1334],
      [1424074500000, "_UNKNOWN_", 93],
      [1424075400000, "3G", 6986],
      [1424880000000, "_UNKNOWN_", 330],
      [1424880900000, "3G", 1213],
      [1424880900000, "_UNKNOWN_", 291],
      [1424881800000, "3G", 1310],
...
      [1424885400000, "_UNKNOWN_", 345],
      [1424886300000, "3G", 1290],
      [1424886300000, "_UNKNOWN_", 366],
      [1424887200000, "3G", 1420],
      [1424889900000, "3G", 669],
      [1424889900000, "_UNKNOWN_", 214]
   ],
   "status": "OK",
   "nbVals": 1734,
   "totalCount": 1734,
   "timeColumn": "EVENT_TIMESTAMP"
}
```

NOTE: Timestamp are returned in EPOC format (also known as POSIX time).
It defined as the number of seconds that have elapsed since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970
You can use some EPOC converter to ease the reading or conversion (http://www.epochconverter.com/)

8.5.2.2 Unified Format (without granularity)

No granularity means the response returned only the aggregated or current values for the time period specified in the request. No timestamp is present in the response.

TIP: Ideal for all graphical representation that display only the current values or status when the request is executed like a dashboard

Example:

http://localhost:9000/V1.0/domains/ossa/packages/MBBQOE_Trial/facts/failed_DNS_req_count,success_DNS_latency_avg/dims/TECHNOLOGY,CONTROLLER?begin=1418688000000&end=1426546799000

will return a column for all dimensions (Technology, Controller) , facts (Failed DNS req count)


```
{
   "headers": ["TECHNOLOGY", "CONTROLLER", "failed_DNS_req_count", "success_DNS_latency_avg"],
   "values": [
      ["3G", "RNC07", 1622, 100.565275608123],
      ["3G", "RNC29", 719, 155.967519133844],
      ["3G", "RNC12", 0, null],
      ["3G", "RNC21", 675338, 65.7292202507793],
      ["_UNKNOWN_", "_UNKNOWN_", 442619, 138.779847830981],
      ["3G", "RNC16", 17, 32.5608108108108],
      ["3G", "RNC30", 0, 5.84615384615385],
      ["3G", "RNC03", 2462490, 136.377153307456],
      ["3G", "RNC25", 0, 19.5251017639077],
      ["3G", "RNC17", 0, 7.07692307692308],
      ["3G", "RNC22", 217816, 85.8449921340997],
      ["3G", "RNC04", 104, 83.8054214630524],
...
   ],
   "status": "OK",
   "nbVals": 31
}
```

8.5.3 Objects / Operations REST API

These URLs manage the object definitions, operations and associated graphical forms for the ObjectType format. This RESTAPI also execute a given create, update, delete operations (CRUD) through the plugin interface.

Description	HTTP Verb	Rest API
Get all object types available in the package	GET	/V1.0/domains/***<domainId>***/packages/<packageId>/types
Get the object type definition in the package	GET	/V1.0/domains/***<domainId>***/packages/***<packageId>***/types /<typeId>/definition
Get all the available forms for a specific object type	GET	/V1.0/domains/***<domainId>***/packages/***<packageId>***/types /<typeId>/forms
Get a specific form definition for an object type in the package	GET	/V1.0/domains/***<domainId>***/packages/***<packageId>***/types /<typeId>/forms/<formId>
Get all the instances of object types in the package	GET	/V1.0/domains/***<domainId>***/packages/***<packageId>***/types /<typeId>/objects
Get the specific instances of the object type in the package	GET	/V1.0/domains/***<domainId>***/packages/***<packageId>***/types /<typeId>/objects<objectId>
Create a new instance of the object type	POST	/V1.0/domains/***<domainId>***/packages/***<packageId>***/types /<typeId>/objects
Update an instance of the object type	PUT	/V1.0/domains/***<domainId>***/packages/***<packageId>***/types /<typeId>/objects/<objectId>
Delete an instance of the object type	DELETE	/V1.0/domains/***<domainId>***/packages/***<packageId>***/types /<typeId>/objects/<objectId>

8.5.4 Self-Monitoring REST API

Plugins needs to provide a RESTAPI to indicate their health status and also be able to customize this api to check all the critical components linked to the plugin (database, external data servers ...).

This API returns a simple response 'OK' that can be easy to parse for a load balancer or any external tools to manage the high availability at plugin level.

Description	HTTP Verb	Rest API
Return the health status 'OK'	GET	/V1.0/domains/***<domainId>***/check

8.5.5 Packages REST API

8.5.5.1 Reload dynamically all packages

The plugin provides a RESTAPI to perform a dynamic reload of all packages associated to a plugin. It will up to date the list of packages available for Unified Console.

Description	
Verb	POST
Path	V1.0/domains/***<domainId>***/reload
Description	Refresh dynamically all the packages associated to the plugin
Path Parameters	• domainId domain identifier
Query Parameters	• reloadWorkspaces=0 to disable reloading workspaces (default is 1)

	• reloadViews=0 to disable reloading views (default is 1) • useStartupConfiguration=0 to ignore config.startup.loadRemoteUIData and config.startup.overwriteRemoteUIData (default is 1)
Response Status Code	200 – OK
Response Object	List of reloaded packages
Errors	

NOTE: This operation may impact the performance of the platform as it may need a little time to be fully completed

Example

This route can be called to refresh dynamically all the packages of a plugin. It reloads all packages (metadata) and all workspaces and views of the plugin, according to startup configuration of the server and query parameters given.

This route must call commons.reloadPackages as its last operation. This common function takes request and response as parameters (see example below). This common function will be in charge of reloading packages, workspaces, views, notifications and notify client users of these reload operations.

Usually the cache of the plugin is cleared, internal data refreshed if needed (workspaces, views, packages, etc.), and then the common function is called to make UOC common services reload the packages by calling the plugin APIs.

Implementation example:

/server/addons/plugins/<domainId>/routes.js

```
app.post(api + '/reload',
accessControl.requireAuthPermissions(['manage_package']),
packageManager.reloadPackages);
```

/server/addons/plugins/<domainId>/package-manager.js

```
...
var fs = require('fs');
...
var plugin = JSON.parse(fs.readFileSync('./plugin.json'));
...
// common functions for plugins, implementations of plugin routes and utility functions
var commonsPluginsUtils = new (require(path.join(process.env.ROOT,
'server', 'addons', 'plugins', 'commons.js')))(plugin);
...
this.reloadPackages = function(req, res) {
    // clear the cache of this plugin / refresh plugin.json, workspaces, views, packages
    // then call this common method that will be in charge of retrieving new workspaces,
    // views, packages, notifications and reload them (update UI database, and notify
    // UOC users)
    commonsPluginsUtils.reloadPackages(req, res);
};
```

8.5.5.2 Reload dynamically a package

Description	
Verb	POST
Path	V1.0/domains/*<domainId>*/packages/<packageId>/reload
Description	Refresh dynamically a package using its unique identifier
Path Parameters	• domainId domain identifier • packageId package identifier
Query Parameters	• reloadWorkspaces=0 to disable reloading workspaces (default is 1) • reloadViews=0 to disable reloading views (default is 1) • useStartupConfiguration=0 to ignore config.startup.loadRemoteUIData and config.startup.overwriteRemoteUIData (default is 1)
Response Status Code	200 – OK
Response Object	List of reloaded packages
Errors	

NOTE: This operation may impact the performance of the platform as it may need a little time to be fully completed

Example

This route can be called to refresh dynamically a specific package of a plugin. It reloads this specific package (metadata) and all its associated workspaces and views coming from the plugin, according to startup configuration of the server and query parameters given.

This route must call commons.reloadPackage as its last operation.

This common function takes request and response as parameters (see example below). This common function will be in charge of reloading the package, its workspaces, views, notifications and notify client users of these reload operations.

Usually the cache part of the plugin corresponding to this package is cleared, internal data refreshed if needed (workspaces, views, package, etc.), and then the common function is called to make UOC common services reload the package by calling the plugin APIs.

Implementation example:

/server/addons/plugins/<pluginId>/routes.js

```
app.post(api + '/packages/:packageId/reload',
accessControl.requireAuthPermissions(['manage_package']),
packageManager.reloadPackage);
```

/server/addons/plugins/<pluginId>/package-manager.js

```
...
var fs = require('fs');
...
var plugin = JSON.parse(fs.readFileSync('./plugin.json'));
...
// common functions for plugins, implementations of plugin routes and utility functions
var commonsPluginsUtils = new (require(path.join(process.env.ROOT,
'server', 'addons', 'plugins', 'commons.js')))(plugin);
...
this.reloadPackage = function(req, res) {
   var packageId = req.params.packagesId; // package identifier
   // clear the cache part of the plugin corresponding to this package / refresh
   // workspaces, views, and metadata of this package if needed.
   // then call this common method that will be in charge of retrieving new workspaces,
   // views, package, notifications and reload them (update UI database, and notify
   // UOC users)
   commonsPluginsUtils.reloadPackage(req, res);
};
```

8.6 Plugin Configuration Files

Configuration files of the plugin that will be place under <install_dir>\server\public\addons\<my plugin Id> directory.

IMPORTANT: Configuration files stored <install_dir>/server/public/<my plugin id> will never be uninstalled during an upgrade. This grant that after an upgrade of the product, there is not lost of configuration, or customization.

Some plugins, like OSS Analytics plugin, have advanced configuration because they manage several data servers that share same northbound interface. So the configuration definition provides multiple servers, protocols, ports definition, etc.

In this case, all the defined servers will be called during startup of UOC to dynamically lookup at packages available.

Example: OSS Analytics Plugin configured to access to 3 different servers : Fault Analytics Statistics, OSS Analytics, and NFV Analytics. These information is used by UOC to query the right server based on requests for a dedicated value pack.

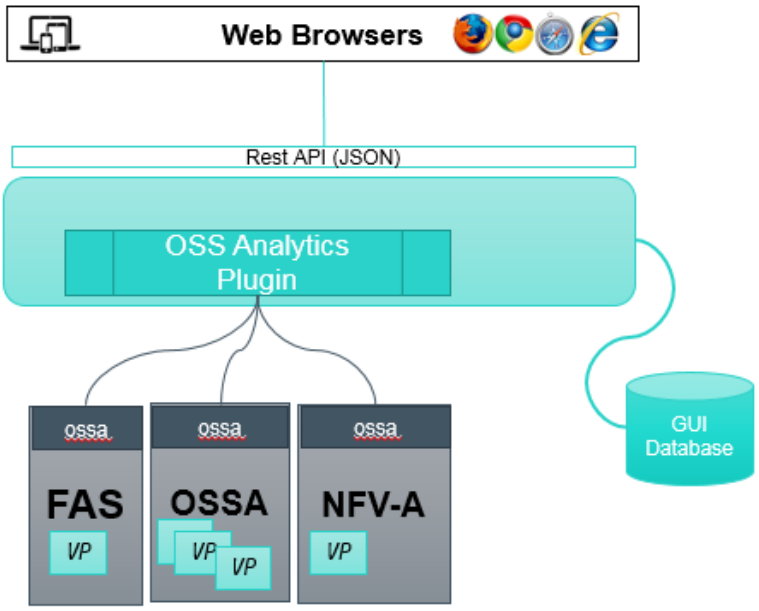


Figure 71 – OSSA Plugin configured for multiple data servers

Example of OSS Analytics Configuration Files

```
{
   "servers": {
            "ossa": {
                     "protocol": "http",
                     "host": "server1.gre.hpe.com",
                     "port": "8080"
            },
            "fas": {
                     "protocol": "http",
                     "host": "server2.gre.hpe.com",
                     "port": "8081"
            },
            "nfva": {
                     "protocol": "https",
                     "host": "server3.gre.hpe.com",
                     "port": "8180"
            }
   },
   "active": true,
}
```

8.7 Role Based Access Control

Add-on Plugin integrates the security model from UOC (Role Based Access Control), and it allows the Add-on developer to control the access of the REST API.

NOTE:. Role Based Access Control (RBAC) is based on roles. A role is a set of permissions. A user can be associated to multiple roles.

Access Control provides 4 levels of route protection:

- no specific control
- the user is correctly authenticated
- the user is correctly authenticated and has the correct roles(s)
- the user is correctly authenticated and has the correct permission(s)

Please refer to 16.2 Access Control API to get the detailed API to protect routes of the plugin (URL)

8.8 Logs and Security Audit

The plugin component usually uses the server logs api for generic server logs (file logs) and the security logger to track access to resources (audit file logs).

Please refer to 16.4 and 16.5 for detailed explanations.

8.9 Integrate an external Node Module in your plugin

This tutorial describes the best solution to integrate a new additional open sources in your plugin. Example with the node module 'uuid' to generate unique identifier. See https://www.npmjs.com/package/uuid

A universal unique identifier (UUID) is a standard of generating identifiers, standardized by the Open Software Foundation (OSF). The intent of UUIDs is to be able to generate unique identifiers in a distributed system without central coordination point. The word unique should not be taken to mean guaranteed unique; it is rather practically unique.

UUID canonical form is as follows: xxxxxxxx-xxxx-Mxxx-Nxxx-xxxxxxxxxxxx

where N indicates the variant and M indicates the UUID version. You can find all the UUID details in RFC4112

Version 1 UUID is meant for generating time-based UUIDs. They also accept 48 bit long identifier (281,474,976,710,655 available values). In many cases it make sense to use a machine MAC Address as an identifier. Sometimes if we use several UUID generators on the same system, we can just use configured identifiers. It is very important to have unique identifier in a distributed environment. That will guarantee conflict-free ids.

Version 4 UUID is meant for generating UUIDs from truly-random or pseudo-random numbers. UUID v4 are not giving us guaranteed unique numbers; they are rather practically unique.

The Node.js node-uuid module is very simple to use. You can install it using npm manager following these steps:

1. Integrate the new dependency in your plugin in the command line
   Npm install uuid **–save**

NOTE: NPM option -save means to persist this changes in the package.json. Without this option, you will need to modify manually this dependencies.

IMPORTANT: This filename 'package.json' is related to the usual filename for NPM command and not related to the Unified Console Package concept.

2. Use this new module in your own code

```
var uuid = require("uuid");
console.log(uuid.v1());
console.log(uuid.v4());
```

Example of console log with UUID:

ec654ec1-7f8f-11e3-ae96-b385f4bc450c

32a4fbed-676d-47f9-a321-cb2f267e2918

Chapter 9 Client Add-ons – Layout

9.1 Overview

The Add-on Layout is in charge of organizing the screen between components and provide a responsive web design to address display constraint on several different devices and its sizes (desktop, laptop, tablet, phone, etc.)

TIP: Responsive web design is an approach to web design aimed at allowing desktop webpages to be viewed in response to the size of the device one is viewing with.

Layout file usually created by a person skilled in web design. It usually require a simple implementation but you need to fully understand technologies like HTML 5, CSS3, Twitter Bootstrap.

It is recommended to follow the mobile first approach and relies on Bootstrap to ease the creation of the layout, especially the Bootstrap grid examples (see http://getbootstrap.com/examples/grid/).

NOTE: Bootstrap is the most popular HTML, CSS and JS framework for developing responsive, mobile first web applications.

See the official Bootstrap web site: http://getbootstrap.com/

By default, UOC provides layouts in its reusable library and you can extend this graphical library with your own layout.

All available layouts installed on the UOC platform can be browse through the user interface if you have the permission to browse layout in your role (platform administrator or view designer by default).

Go to the default menu-bar, select Add-ons, then Layouts.

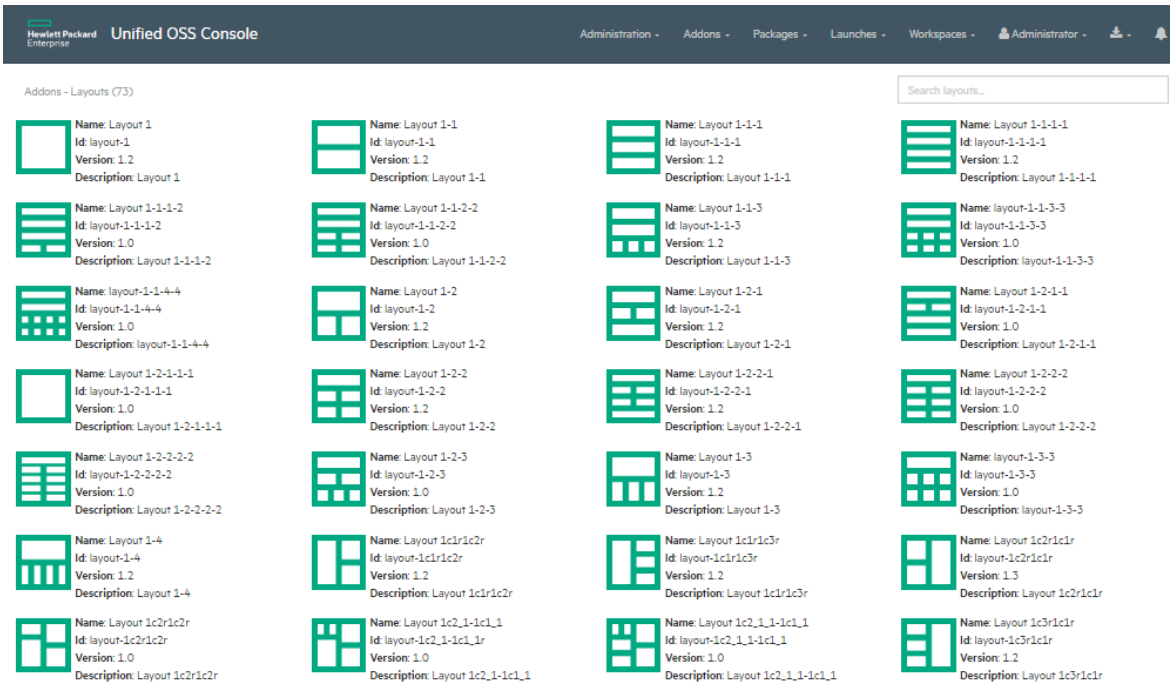


Figure 72 – Add-ons – Layouts Management

Layouts can be found under <install_dir>/clients/addons/<addonId>/layouts

9.2 Structure

To create a new layout, it is strongly recommended to use the SDK generator to generate a working skeleton at the right location based on your Add-on identifier.

Layout is placed under <install_dir>\client\addons\<my add-on Id>\layouts\<my add-on id>-<my layout id> directory and will be composed of the following files:

Files	Description
layouts.json	Descriptor of the layout Note: The name is fixed and cannot be changed.
<my add-on id>-<my-layout id>.html	Implementation of the layout as HTML file based on HTML, CSS and bootstrap.

It is recommended to have one directory per layout to ease maintenance and reusability.

9.3 Descriptor

Descriptor is a mandatory file because it is used by UOC to identify the layout, and associate to this layout which HTML template to use during dynamic screen generation by UOC graphical engine. It also contains all information associated to this layout (presentation e, version, author, etc.)

Please refer to the object JSON definition in 6.15 Layout

Layout descriptor is under <install_dir>\client\addons\<my add-on Id>\layouts\<my add-on id>-<my layout id>\layouts.json

Example of descriptor for layout 2-2-1 (addon hpe) that contains 5 widgets

```
[{
  "layoutId": "layout-2-2-1",
  "name": "Layout 2-2-1",
  "description": "Layout 2-2-1",
  "author": "hpe",
  "version": "1.2",
  "icon": "layout-2-2-1.png",
  "categories": ["HPE"],
  "domains": [],
  "packages": [],
  "widgetCount": 5,
  "responsive": true,
  "templateUrl": "/addons/hpe/layouts/layout-2-2-1/layout-2-2-1.html"
}]
```

9.4 Layout File

The layout file is a HTML file that describe the HTML tags, CSS style and bootstrap class (responsive, style…).

A layout format is very closed to HTML and it can integrate static information and dynamic information relying on add-ons widgets. Layout are defined without any strong coupling with predefined widgets. A generic 'space' for widget is specified and the graphical engine will be in charge of injecting the right widgets and its configuration at the right place in the layout.

With such definition, the layout is generic and can be reused whatever is the data to display.

IMPORTANT: Each available space for widget is defined using the generic HTML tag '**hpe-widget**'.

Example of layout 2-2-1 that contains 5 widgets: layout-2-2-1.html

```
<div>
  <div class="row">
    <div class="col-md-6">
      <hpe-widget index="0" context="context"></hpe-widget>
    </div>
    <div class="col-md-6">
      <hpe-widget index="1" context="context"></hpe-widget>
    </div>
  </div>
  <div class="row">
    <div class="col-md-6">
      <hpe-widget index="2" context="context"></hpe-widget>
    </div>
    <div class="col-md-6">
      <hpe-widget index="3" context="context"></hpe-widget>
    </div>
  </div>
  <div class="row">
    <div class="col-md-12">
      <hpe-widget index="4" context="context"></hpe-widget>
    </div>
</div>
```

1st column
2nd column
1st row
1st column
2nd column
2nd row
1st column
3rd row

IMPORTANT: index of the hpe-widget starts at zero. Index is used to find a in the Views definition (array of widgets configuration)

9.5 Icon

By default, the SDK generator does not generate a preview icon. You can manually customize a specific icon and copy it in the following directory following the naming convention:

<install_dir>\client\public\images\layouts\<my add-on id>-<my-layout id>.png

Default format is PNG and default size is 256x256.

IMPORTANT: To support theming, it is recommended to make this icon transparent (background color will apply the color to the layout)

Chapter 10 Client Add-ons – Widget

10.1 Overview

A widget is a graphical reusable component that can be displayed on views. It uses a predefined space on a layout identified by **hpe-widget** HTML tag (see 9.4 Layout File).

UOC provides a wide set of widgets. These widgets support the unified formats defined in a value pack. So, all these widgets can be reused for different use cases even if data come from different value packs.

By default, UOC provides widgets in its reusable library and you can extend this graphical library with your own widget.

All available widgets installed on the UOC platform can be browse through the user interface if you have the permission to browse widget in your role (platform administrator or view designer by default).

Go to the default menu-bar, select Add-ons, then Widgets.

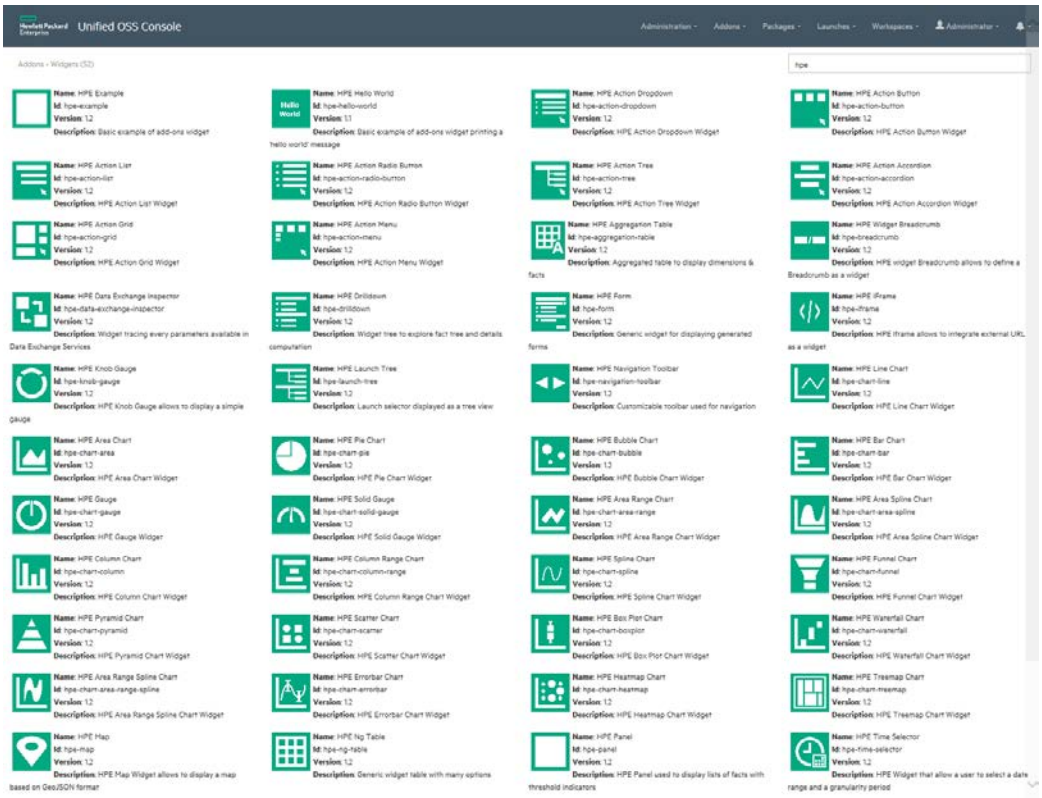


Figure 73 – Add-ons – Widgets Management

NOTE: The View Designer will display all the available widgets found in the widget library.

There are several types of widgets:

- Selection: A Selection widget is dedicated to capture the input of the user, validate this input, and publish this information to other widgets (using the supported output parameters of the widget). Best examples are widget action, widget search, time selector, etc.
- Visualization: A visualization widget is in charge of presenting information retrieved from plugins. It displays metrics, status, colors, based on the results of queries from the data server. Best examples are table, charts, gauge, heat map, etc.
- Containers: A container widget is in charge of managing child widgets. The widget container acts as a parent and it is usually used to group widgets or to create a sort of cinematic between them.

- Widget Group: A widget group is a container (it groups child widgets and share the same data selection for queries).
- Widget Navigation: This widget manages a set of child widgets and select one at a time to be active and visible. For example, it can be used to display first a list of objects in a widget 'list' allowing clicking on a row to select the item of the list and display its details with another widget 'detail'. Only the active widget is visible and all child widgets of a Widget Navigation share the same area of the view.
- Widget Tab: This widget manages a set of child widgets as tabular window.

- Hybrid (Visualization & Selection): This kind of widget is usually used to display some synthetic information or lists of metrics, locations,... They allow also a selection to act as a filter or selector, usually to get more information about the selected item(s). For example, it can be a colored map displaying the region with specific status (high, medium, warning...) and each region can be selected to apply a filter to other widgets and focus on the result of this specific region.

10.2 Default HPE Widgets

By Default HPE provides several widgets in its add-ons. Please read the UOC Integration Guide to get details and description for each of them.

Widget Identifier	Name
hpe-chart-line	HPE Line Chart
hpe-chart-area	HPE Area Chart
hpe-chart-pie	HPE Pie Chart
hpe-chart-bubble	HPE Bubble Chart
hpe-chart-bar	HPE Bar Chart
hpe-chart-area-range	HPE Area Range Chart
hpe-chart-area-spline	HPE Area Spline Chart
hpe-chart-column	HPE Column Chart
hpe-chart-column-range	HPE Column Range Chart
hpe-chart-spline	HPE Spline Chart
hpe-chart-funnel	HPE Funnel Chart
hpe-chart-pyramid	HPE Pyramid Chart
hpe-chart-scatter	HPE Scatter Chart
hpe-chart-boxplot	HPE Box Plot Chart
hpe-chart-heatmap	HPE Heatmap Chart
hpe-chart-treemap	HPE Treemap Chart
hpe-chart-gauge	HPE Gauge Chart
hpe-chart-solid-gauge	HPE Solid Gauge Chart
hpe-knob-gauge	HPE Knob Gauge Chart
hpe-aggregation-table	HPE Aggregation Table
hpe-data-exchange-inspector	HP Data Exchange Inspector
hpe-iframe	HPE IFrame
hpe-time-selector	HPE Time Selector
hpe-top-table	HPE Top Table
hpe-drilldown	HPE Drilldown
hpe-widget-group (*)	HPE Widget Group

hpe-widget-navigation (*)	HPE Widget Navigation
hpe-form	HPE Widget Form
hpe-ng-table	HPE Widget Table
hpe-widget-menu-bar	HPE Widget Menu Bar
hpe-breadcrumb	HPE Widget Breadcrumb
hpe-map	HP Widget Map
hpe-tree-launch	HPE Tree Launch
hpe-action-menu	HPE Action Menu
hpe-action-list	HPE Action List
hpe-action-tree	HPE Action Tree
hpe-action-grid	HPE Action Grid
hpe-action-button	HPE Action Button
hpe-action-radio-button	HPE Action Radio Button
hpe-action-dropdown	HPE Action Drop Down
hpe-action-accordion	HPE Action Accordion
Hpe-action-toggle	HPE Action Toggle
hpe-search	HPE Search
hpe-ng-leaflet	HPE Leaflet
hpe-card	HPE Card
hpe-meter	HPE Meter
hpe-widget-tab (*)	HPE Widget tab
hpe-localization-inspector	HPE Localization Inspector
hpe-notifications-table	HPE Notifications Table
hpe-notifications	HPE Notifications
hpe-notifications-generator	HPE Notifications Generator
hpe-notifications-keywords	HPE Notifications Keywords
hpe-chat	HPE Chat

Table 2 –Default HPE Widgets

(*) These widgets are container for other widgets to provide advanced features to child widgets (animation, information sharing…)

NOTE: Default HPE widgets are under <install_dir>/client/addons/hpe/widgets

10.3 Structure

Widget will be placed under <install_dir>\client\addons\<my add-on Id>\widgets\<my add-on id>-<my widget id> directory and will be composed of the following files:

Files	Description
widgets.json	Descriptor of the widget Note: The name is fixed and cannot be changed.
<my add-on id>-<my widget id>.json	JSON Schema to configure this widget
<my add-on id>-<my widget id>.tpl.html	Definition of the template HTML to describe the HTML new tag to use. Naming convention: postfix this file by –tpl.html
<my add-on id>-<my widget id>.html	Implementation of the widget as HTML file based on bootstrap.
<my add-on id>-<my widget id>.init.html	Implementation of the widget as HTML file based on bootstrap. Naming convention: postfix this file by –init.html
<my add-on id>-<my widget id>.cfg.html	Implementation of the widget configuration popup content as HTML file based on bootstrap. Naming convention: postfix this file by –cfg.html
<my add-on id>-<my widget id>.css	Implementation of the widget style as CSS.
<my add-on id>-<my widget id>.js	Implementation of the main module for the widget
<my add-on id>-<my widget id>-controllers.js	Implementation of controllers for the widget Naming convention: postfix this file by –controllers.js
<my add-on id>-<my widget id>-controllers-test.js	Controllers unit test file validating that controllers can be loaded without error. (if tests were generated) Naming convention: postfix this file by –controllers-test.js
<my add-on id>-<my widget id>-directives.js	Implementation of directives for the widget Naming convention: postfix this file by –directives.js
<my add-on id>-<my widget id>-directives-test.js	Directives unit test file validating that directives can be loaded without error. (if tests were generated) Naming convention: postfix this file by –directives-test.js
<my add-on id>-<my widget id>-services.js	Implementation of services for the widget Naming convention: postfix this file by –services.js
<my add-on id>-<my widget id>-services-test.js	Services unit test file validating that services can be loaded without error. (if tests were generated) Naming convention: postfix this file by –services-test.js
<language code>.json	One localization JSON file per supported language codes. *(e.g. en-us.json, fr-fr.json,...)*

10.4 Descriptor

The descriptor is mandatory and defines the widget object. Please refer to the object JSON definition in 6.14 Widgets

It basically indicate:

- the registered identifier for this widgets
- the HTML template file to use during the rendering of the widget,
- the HTML file to use for configuration panel
- the HTML file to use for initialization panel
- default widget toolbar button preferences
- the list of input parameters
- the list of output parameters

Example of descriptor for widget

```
[{
   "widgetId": "xxx-my-widget",
   "name": "My Widget",
   "description": "This is my widget",
   "author": "xxx",
   "version": "1.0",
   "icon": "default.png",
   "categories": ["xxx"],
   "domains": [],
   "packages": [],
   "templateUrl": "/addons/xxx/widgets/xxx-my-widget/xxx-my-widget.tpl.html",
   "optionsTemplateUrl": "/addons/xxx/widgets/xxx-my-widget/xxx-my-widget.cfg.html",
   "initTemplateUrl": "/addons/xxx/widgets/xxx-my-widget/xxx-my-widget.init.html",
   "toolbar": {
      "selectors": true,
      "refresh": true,
      "zoom": false,
      "configuration": true,
      "launch": false
   },
   "inputs": [{
      "id": "TIMERANGE_FROM",
      "name": "FROM",
      "description": "Beginning of the selected time period"
   }, {
      "id": "TIMERANGE_TO",
      "name": "TO",
      "description": "End of the selected time period"
   }, {
      "id": "GRANULARITY",
      "description": "Granularity value in seconds",
      "name": "GRANULARITY"
   }, {
      "id": "WIDGET_SELECTIONS",
      "description": "Data selection parameter to populate on selection"
   }],
   "outputs": [{
      "id": "URL",
      "description": "URL to display inside the Widget IFrame (internal launch)"
   }, {
      "id": "WIDGET_SELECTIONS",
      "name": "SELECTIONS",
      "description": "Data selection parameter to populate on selection"
   }]
}]
```

10.5 JSON Schema

JSON Schema is a vocabulary that allows you describe the existing data format and ease the validation of JSON documents. The latest IETF published draft is v4.

NOTE: Please refer to JSON schema specification and JSON schema (http://json-schema.org)

This JSON Schema describes the configuration section of the widget and the allowed JSON syntax to use while configuring a view without the graphical configuration panel.

By default, it describe the title (text and color), show or hide title, size (width and height) and initialization section.

```
{
   "$schema": "http://json-schema.org/draft-04/schema#",
   "type": "object",
   "title": "My Widget",
   "description": "This is my widget",
   "properties": {
      "configuration": {
         "type": "object",
         "title": "Configuration",
         "description": "Configuration of xxx-my-widget",
         "properties": {
            "title": {
               "type": "string",
               "title": "Title",
               "description": "Title of xxx-my-widget"
            },
            "titleColor": {
               "type": "string",
               "title": "Title color",
               "description": "Title color (red, #FF0000...)"
            },
            "showTitle": {
               "type": "boolean",
               "title": "Show title",
               "description": "Show or hide the title"
            },
            "width": {
               "type": "string",
               "title": "Width",
               "description": "Width (300px, ...)"
            },
            "height": {
               "type": "string",
               "title": "Height",
               "description": "Height (300px, ...)"
            },

            "init": {
               "type": "object",
               "title": "Initialization",
               "description": "Initialization of xxx-my-widget",
               "properties": { },
               "required": []
            }
         }
      }
   }
```

```
}
```

10.6 Directives File

Please refer to 2.8 Angular Directives for detailed explanation about directives.

This file describe exactly how to render the custom tag using standard HTML (templateURL).

```
define([
    'angular',
    'addons/xxx/widgets/xxx-my-widget/xxx-my-widget-controllers',
    'css!addons/xxx/widgets/xxx-my-widget/xxx-my-widget.css'
], function(angular) {
    'use strict';
    var xxxMyWidgetDirectives = angular.module('xxxMyWidgetDirectives',
['xxxMyWidgetControllers']);

    xxxMyWidgetDirectives.directive('xxxMyWidget', [
        function() {
            return {
                restrict: 'E',
                replace: true,
                scope: {
                    widget: '=',
                    context: '='
                },
                controller: 'xxxMyWidgetController',
                templateUrl: 'addons/xxx/widgets/xxx-my-widget/xxx-my-
widget.html'
            };
        }
    ]);

    return xxxMyWidgetDirectives;
});
```

10.7 Template File

The template file (*.tpl.html) is a file specific to UOC that in charge of describing the HTML extension available (i.e. angular directive) to use by the UOC graphical engine to generate the widget content in a view..

```
<xxx-my-widget widget="widget" context="context"></xxx-my-widget>
```

By default, the widget template inject in the widget scope, all the needed information define for this widget i.e. widget configuration described in the view, and the context of execution (workspace id, view id, etc.) for advanced customization.

Example with training-weather widget

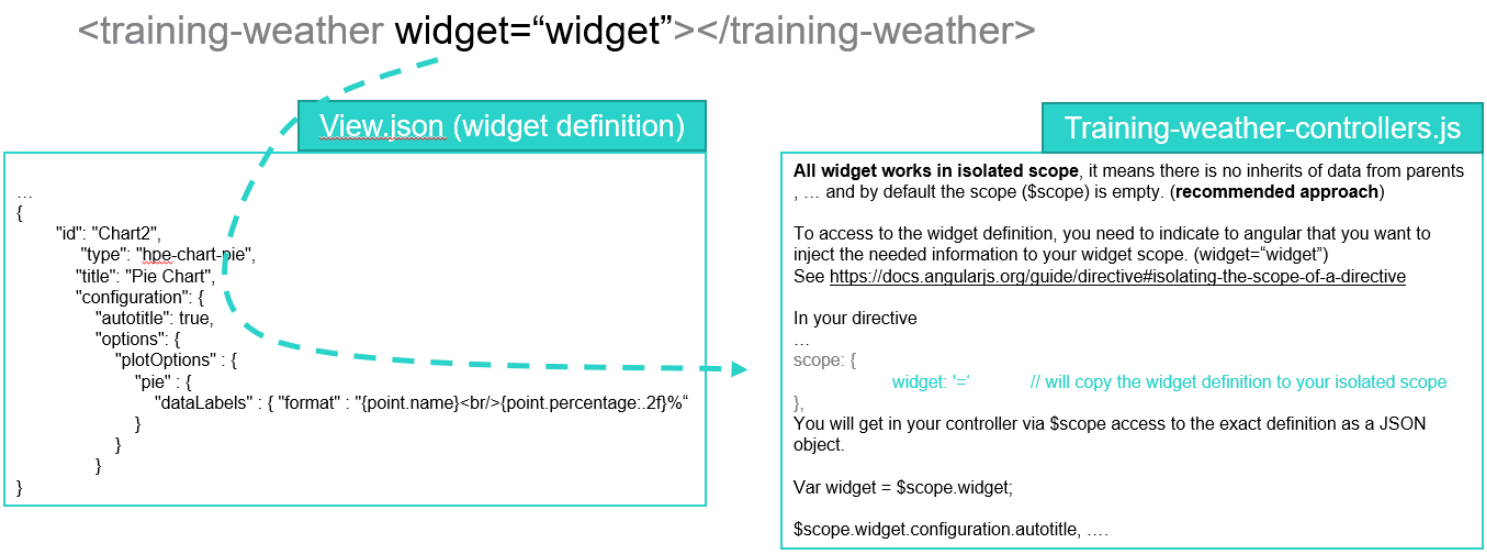


Figure 74 – Add-ons – Widgets Template file (widget attribute)

Please refer to 2.8 Angular Directives and 2.9 Angular Scopes to get detailed explanation.

10.8 HTML File

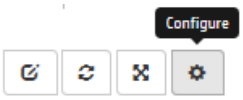
The HTML file implement the HTML rendering to use after replacing the custom tag (directive). This file follow the HTML 5 / CSS 3 standard and usually leverage bootstrap to provide a responsive design to the web page. It also have to take care of the theming support provided by UOC.

By default, the code generator generates a widget that display the title (if any) a TODO and "no data" text and support localization. it integrate support for angular translate through the directive **<translate>**

```
<div class="xxx-my-widget">
    <h1 ng-show="widget.configuration.title &amp;&amp;
widget.configuration.showTitle !== false"
        data-ng-style="{color: widget.configuration.titleColor}">
        {{widget.configuration.title}}
    </h1>

    <!-- xxx-my-widget Code here -->
    <p class="text-center">TODO xxx-my-widget</p>
```

```
    <p data-ng-if="!data" class="text-
center"><strong><translate>XXX_MY_WIDGET.NO_DATA</translate></strong></p
>
    <div data-ng-if="data">
        <h2>data:</h2>
        <samp data-ng-if="data">
            {{data}}
        </samp>
    </div>
</div>
```

10.9 HTML Configuration File



The HTML Configuration file is used by UOC when the user click to the button “configure”.

Two options are available for an add-on developer: implement a user friendly graphical panel for the configuration or let UOC displays the JSON editor.

- If the add-ons developer implemented a graphical configuration panel (e.g. form), this html is displayed in a popup dialog to let the end user modify the confirmation properties. In this case, the descriptor (optionsTemplateUrl) must define the file to associate and this HTML file must be implemented.
- If the add-ons developer do not want to provide a pre-define GUI for configuration, he does not have to associate a file to optionsTemplateUrl property in the descriptor and does not have to implement a HTML configuration file.

NOTE: The View Designer will let you always access to the JSON Editor and the configuration dialog box.

```
<form class="form-horizontal xxx-my-widget-configuration" data-ng-
controller="xxxMyWidgetConfigurationController">
    <!--
    This file is the content of the widget configuration modal.
    It allows to modify some widget configuration options (public
configuration).
    This configuration modal will be available to the end user, and in
the View Designer.
        The only variable available in the scope of this form is:
widgetConfiguration
    All properties of widgetConfiguration will be applied to
widget.configuration once the apply button is clicked.
    -->
    <div class="row">
        <div class="col-sm-6">
            <div class="form-group">
                <label for="xxxMyWidgetConfigTitle" class="col-sm-4
control-label"><translate>XXX_MY_WIDGET.CONFIG.TITLE</translate></label>
                <div class="col-sm-8">
```

```
                    <input id="xxxMyWidgetConfigTitle" type="text"
placeholder="{{'XXX_MY_WIDGET.CONFIG.TITLE'|translate}}" class="form-
control" name="xxxMyWidgetConfigTitle" ng-
model="widgetConfiguration.title">
                </div>
            </div>
            <div class="form-group">
                <label for="xxxMyWidgetConfigTitleColor" class="col-sm-4
control-
label"><translate>XXX_MY_WIDGET.CONFIG.TITLE_COLOR</translate></label>
                <div class="col-sm-8">
                    <div class="input-group">
                        <input id="xxxMyWidgetConfigTitleColor"
type="color" class="form-control" ng-
model="widgetConfiguration.titleColor"
tooltip="{{'XXX_MY_WIDGET.CONFIG.TOOLTIP.COLOR'|translate}}">
                        <span class="input-group-btn">
                            <button class="btn btn-default"
type="button" data-ng-click="clear(widgetConfiguration, 'titleColor')"
tooltip="{{'XXX_MY_WIDGET.CONFIG.CLEAR' | translate}}"><i class="fa fa-
eraser"></i></button>
                        </span>
                    </div>
                </div>
            </div>
        </div>
        <div class="col-sm-6">
            <div class="form-group">
                <label for="xxxMyWidgetConfigShowTitle" class="col-sm-4
control-label"
tooltip="{{'XXX_MY_WIDGET.CONFIG.TOOLTIP.SHOW_TITLE'|translate}}">

<translate>XXX_MY_WIDGET.CONFIG.SHOW_TITLE</translate>
                </label>
                <div class="col-sm-8">
                    <div class="checkbox">
                        <input id="xxxMyWidgetConfigShowTitle"
type="checkbox" ng-model="widgetConfiguration.showTitle">
                    </div>
                </div>
            </div>
        </div>
    </div>
    <hr>
    <div class="row">
        <div class="col-sm-6">
```

```
            <div class="form-group">
                <label for="xxxMyWidgetConfigWidth" class="col-sm-4
control-label"><translate>XXX_MY_WIDGET.CONFIG.WIDTH</translate></label>
                <div class="col-sm-8">
                    <input id="xxxMyWidgetConfigWidth" type="text"
placeholder="{{'XXX_MY_WIDGET.CONFIG.WIDTH'|translate}}" class="form-
control" name="xxxMyWidgetConfigWidth" ng-
model="widgetConfiguration.width">
                </div>
            </div>
        </div>
        <div class="col-sm-6">
            <div class="form-group">
                <label for="xxxMyWidgetConfigHeight" class="col-sm-4
control-
label"><translate>XXX_MY_WIDGET.CONFIG.HEIGHT</translate></label>
                <div class="col-sm-8">
                    <input id="xxxMyWidgetConfigHeight" type="text"
placeholder="{{'XXX_MY_WIDGET.CONFIG.HEIGHT'|translate}}" class="form-
control" name="xxxMyWidgetConfigHeight" ng-
model="widgetConfiguration.height">
                </div>
            </div>
        </div>
    </div>
</form>
```

Example of training-weather HTML configuration file

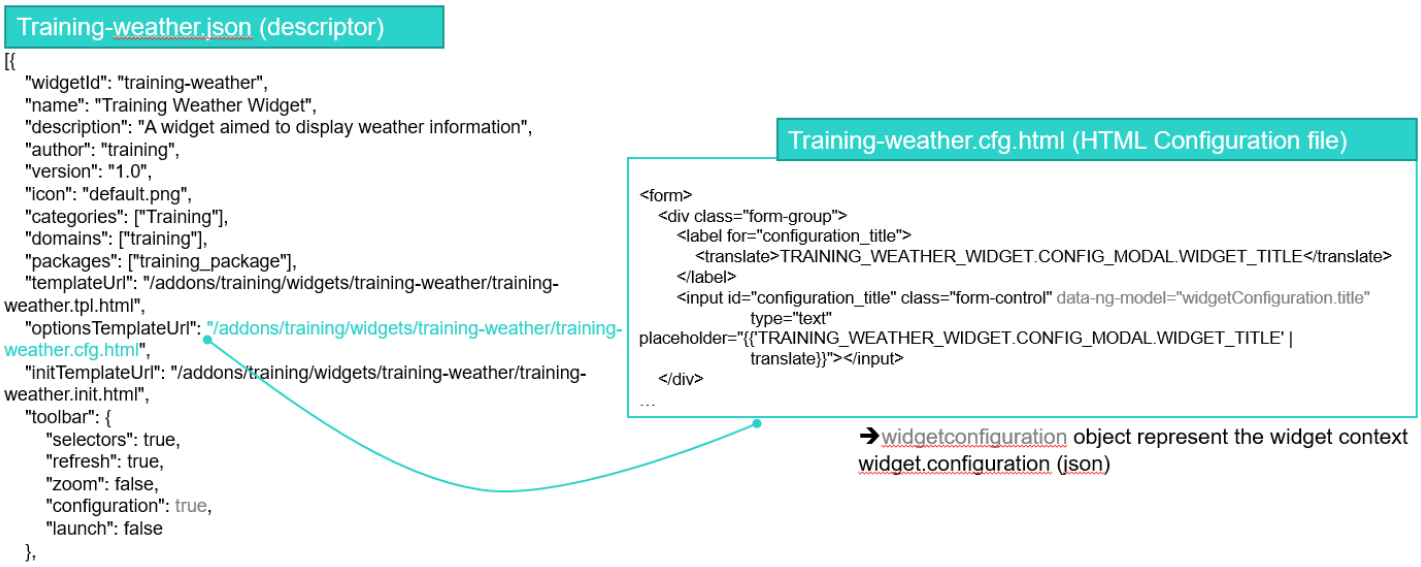


Figure 75: Example of training-weather HTML configuration file

10.10 HTML Initialization File

The HTML Initialization file is used by UOC when the user click to the button “initialization in the View Designer only. Two options are available for an add-on developer: implement a user friendly graphical panel for the initialization or let UOC displays the JSON editor.

- If the add-ons developer implemented a graphical initialization panel (e.g. form), this html is displayed in a popup dialog to let the end user modify the confirmation properties. In this case, the descriptor (initTemplateUrl) must define the file to associate and this HTML file must be implemented.
- If the add-ons developer do not want to provide a pre-define GUI for initialization, he does not have to associate a file to initTemplateUrl property in the descriptor and does not have to implement a HTML initialization file.

NOTE: The View Designer will let you always access to the JSON Editor and the initialization dialog box.

By default, the code generator generates an empty initialization panel indicating this panel need to be customized.

```
<form class="form-horizontal xxx-my-widget-initialization" role="form">
    <!--
    This file is the content of the widget initialization modal.
    It allows to initialize some widget internal configuration in
opposition to the configuration modal (public configuration) (xxx-my-
widget.cfg.html).
    This configuration modal will be used by the View Designer in order
to configure a given instance of this type of widget.

    The only variable available in the scope of this form is:
widgetConfiguration
    All properties of widgetConfiguration will be applied to
widget.configuration once the apply button is clicked.
    -->
    <div class="form-group">
        <div class="col-sm-12">
            <p class="text-center">This widget has no initialization
option</p>
        </div>
    </div>
</form>
```

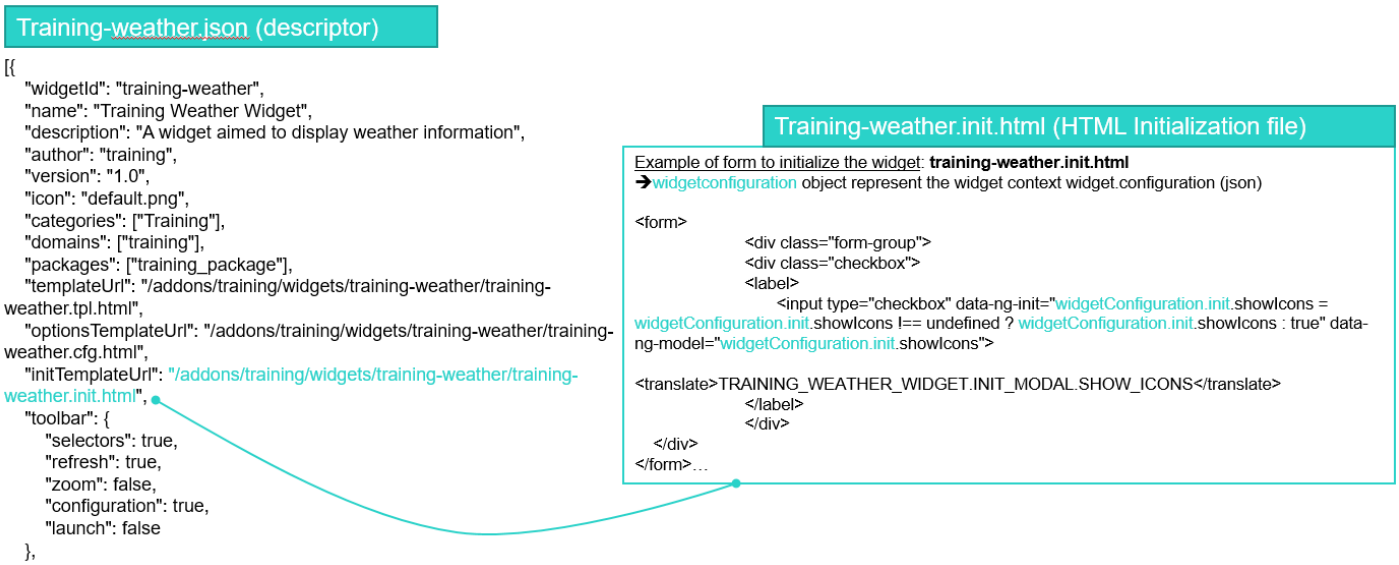


Figure 76: Example of training-weather HTML initialization file

10.11 CSS File

The CSS file implement the style associated to the HTML classes for your widget, configuration and your initialization panels. This is important to isolate the CSS of this module, to avoid breaking the display of other parts of the product. It is a common mistake to customize style that have a global impact. So, it is mandatory to follow this syntax:

```
/*
    This file contains the CSS code associated to the widget xxx-my-widget.
    All the CSS code in this file should be like:

    .xxx-my-widget a-CSS-selector {
        CSS rules
    }
    This is important to isolate the CSS of this widget, to avoid breaking the display of other parts of the product.
*/
.xxx-my-widget {
}

/*
 * Configuration modal window
 */
.xxx-my-widget-configuration label, .xxx-my-widget-configuration input[type=checkbox] {
    cursor: pointer;
}

/*
```

```
 * Initialization modal window
 */
.xxx-my-widget-initialization {
}
```

10.12 Controllers File

Please refer to 2.10 Angular Controllers for detailed explanation about controllers.

10.12.1 Basic implementation

After using the SDK code generator to build a first working skeleton, a controller and its logger are available for implementation .

Several services are already available in the controllers module:

- Common events. (see 18.1 Common Events)
- Theme management. (see 18.9 Theme Configuration Services)
- Data exchange services. (see 18.2 Data Exchange Services)
- Data access services. (see 18.3 Data Access Services)

And the generated widget services.

10.12.2 Theme Configuration Support

The Widget leverages an advanced features provided by the add-on theme and the UOC graphical engine while rendering the view. All details of this processing can be read in 13.3 Theme Configuration Support for Components.

IMPORTANT: the SDK generate does not provide an empty skeleton in the UOC themes to prepare any specific customization. It is in charge of the add-on developer to add this file and CSS specific file to support this advanced feature

The widget developer provides a default style for its widget in the widget module and need to integrate the Theme Configuration Services and init it correctly in its controller to support a dynamic customization driven by the add-on theme (done by other integrators, external teams, customers, etc.)

By default, HPE dark and light themes integrates many customization on widgets to follow the HPE branding.

There is a strict naming convention to make sure the UOC graphical engine associate the right CSS file to the right component during the rendering step.

Component Identifier must be unique leverage the add-on identifier

```
<my add-on id>-<my widget id>.css
```

In

```
<install dir>/client/addons/<theme id>/themes/css
```

Example of Theme Configuration Support in a widget

```
define([
      'commons/events/commons-events',
      'components/theme-config/theme-config-services'
],
function(angular) {
var hpeFormControllers = angular.module('hpeFormControllers', [
'commonsEvents', 'themeConfigServices']);
      hpeFormControllers.controller('hpeFormController', ['events',
function(events, themeConfigService) {
      themeConfigService.applyThemeCSS('hpe-form');
…
// We listen a specific event 'themeChanged' to manage the dynamic
change of theme by end user.
       $scope.$on(events['webgui.themeChanged'], function() {
                          themeConfigService.applyThemeCSS('hpe-form');
                    });
```

NOTE: applyThemeCSS will check in the theme CSS directory if a specific customization for this widget has been done for this theme and if it exists, it overrides the default CSS style implemented by the developer.

More details in 18.9 Theme Configuration Services and 0

webgui.themeChanged

10.12.3 Widget events

This controller implement by default all needed JavaScript event to manage the behavior of the widget.

The add-on developer can listen and catch these events to customize the processing at various points of the widget lifecycle:

webgui.widgetRefresh : Triggered when the widget must be refreshed, for example when clicking on the button refresh of the widget's toolbar.

- webgui.widgetRefresh
- webgui.widgetCancelRefresh: Triggered when the widget refresh must be canceled, for example when clicking on the button cancel of the widget's toolbar. See 18.1.4 webgui.widgetCancelRefresh

webgui.widgetResize: Triggered when the size of the view changes, for example when opening the data selection menu. It is not triggered on window resize, only on some specific cases.

- webgui. widgetResize

webgui.widgetConfigurationChanged: Triggered when the configuration of a widget has been changed using the configuration dialog.

- webgui.widget.configurationChanged

webgui.themeChanged: Triggered when the UOC theme has been changed.

- webgui.themeChanged. A typical usecase is to propaget some changes on embedded open sources that manage themes in their own way.

These events are defined in **commons-events** module ("**commonsEvents**").

```
        var events = {
                // identifier to use : string to exchange at runtime (should be the same)
                'webgui.widgetRefresh': 'webgui.widgetRefresh',
                'webgui.widgetCancelRefresh': 'webgui.widgetCancelRefresh',
                'webgui.widgetResize': 'webgui.widgetResize',
                'webgui.dataExchange': 'webgui.dataExchange',
                'webgui.widgetConfigurationChanged': 'webgui.widgetConfigurationChanged',
                'webgui.themeChanged': 'webgui.themeChanged',
...
        };
```

10.12.4 Autoload feature

By default, a widget is instanciated in the layout, and start to retrieve its associated data with queries (i.e. autoload feature is true). Some widgets may choose to turn off the autoload property to force the end user to manually click on refresh button to start to collect data and process them for display. IT could be interesting in some cases when the data collection and processing is long or impact too much performance at data server level).

There is an important variable named "**autoload**" that by default, send an event to start an initial refresh and make sure that as soon as the widget is loaded, the widget need to collect data and display them. This is the default behavior is this property in the widget configuration is not run off explicitly.

Example of widget configuration to turn off the autoload feature:

```
    {
      "id": "AggregationTable",
      "type": "hpe-aggregation-table","title": "Aggregation Table",
      "autoload": false,
      "configuration": {}
}
```

The widget developer has to manage this standard option of UOC Widget with this simple piece of code in the controller.

```
...
// Support autoload option (can be set to true in the widget definition
in the view). Usually triggers a refresh. default: true
   if ($scope.widget.autoload !== undefined ? $scope.widget.autoload :
true) {
          // wait for the digest to be over
          // (if this $timeout is removed, the redraw will be triggered
before the widget uniqueId has been populated)
          $timeout(function() {
               refresh();
           }, 0);
       }
...
```

10.12.5 Logs

UOC relies on the angular logs service ($log) for add-ons client and to open the web browser console of your favorite browser, you usually can press F12. See 2.13 Angular Logs

Example of $log usage in UOC

```
trainingWeatherControllers.controller('trainingWeatherController',
['$scope', '$log',
    function($scope, $log) {
      var logger = $log.getInstance('trainingWeatherController');
…
      logger.error('An error occurred while retrieving data from server:
', err);
```

By default, the $log is generated by the SDK code generator and it is only recommended to use it for your own logs.

IMPORTANT: These logs are not persisted or send to the server side for storage. Goal is to avoid issue with performance and avoid useless request between the web browser and the UOC server.

10.12.6 Message and notifications

Widget totally relies on the common client service message-notifier to send visual message to the notification area. The SDK code generator integrates by default this service and you only need to leverage its API for your own need in your widget. Please refer to 18.10 Message Notifier Services.

Example of message-notifier services in the widget training-weather:

```
define([
…
'components/message-notifier/message-notifier-services',
…
], function(angular)
{
     'use strict';
    var trainingWeatherControllers =
angular.module('trainingWeatherControllers',
['messageNotifierServices']);

    trainingWeatherControllers.controller('trainingWeatherController',
['$scope', '$log', 'messageNotifierService',
    function($scope, $log, messageNotifierService) {
…
                  messageNotifierService.error
('TRAINING_WEATHER_WIDGET.MESSAGES.ERRORS.GET_DATA_ERROR');
…
};
```

- Services File

```
define([
    'angular',
    'lodash',
    'commons/events/commons-events',
    'components/message-notifier/message-notifier-services',
    'components/data-exchange/data-exchange-services',
    'components/data-access/data-access-services',
    'components/theme-config/theme-config-services',
    'addons/xxx/widgets/xxx-my-widget/xxx-my-widget-services'
], function(angular, _) {
    'use strict';
    var xxxMyWidgetControllers =
angular.module('xxxMyWidgetControllers', ['commonsEvents',
'messageNotifierServices', 'dataExchangeServices', 'dataAccessServices',
'themeConfigServices', 'xxxMyWidgetServices']);


    xxxMyWidgetControllers.controller('xxxMyWidgetController',
['$scope', '$log', '$timeout', 'events', 'messageNotifierService',
'dataExchangeService', 'dataAccessService', 'themeConfigService',
'xxxMyWidgetService',
        function($scope, $log, $timeout, events, messageNotifierService,
dataExchangeService, dataAccessService, themeConfigService,
xxxMyWidgetService) {


            // client-side enriched angular logger
            var logger = $log.getInstance('xxxMyWidgetController');


            // init the widget according to its configuration
($scope.widget.configuration and $scope.widget.configuration.init)
            if(_.get($scope, 'widget.configuration')) {
                logger.debug('Initialization: loading widget
configuration');
                //
                if($scope.widget.configuration.init) {
                    logger.debug('Initialization: loading widget
configuration.init');
                    //
                }
            }
            // function called  when the data has changed and a redraw
of the widget is needed
            var redraw = function(newData) {
                //
                $scope.data = newData;
```

```
            };

            // Refresh function, called each time data need to be
updated
            var refresh = function() {
                $scope.widget.busy = true;

xxxMyWidgetService.getData($scope.widget).then(function(data) {
                    redraw(data);
                }).catch(function(err) {
                    logger.error('An error occurred while retrieving
data', err);

messageNotifierService.error('XXX_MY_WIDGET.MESSAGES.ERRORS.GET_DATA_ERR
OR');
                }).finally(function() {
                    $scope.widget.busy = false;
                });
            };

            // functions called when an input changes (key = input.id)
            // Should be extended as needed
            var onInputChange = {
                TIMERANGE_FROM: function(input) {
                    return onInputChange.WIDGET_SELECTIONS(input);
                },
                TIMERANGE_TO: function(input) {
                    return onInputChange.WIDGET_SELECTIONS(input);
                },
                WIDGET_SELECTIONS: function(input) {
                    logger.debug($scope.widget.uniqueId, 'Input changed:
', input.name);
                    refresh();
                }
            };

            // listen to changes of inputs with "listen = true"
            var listenToInputs = function() {
                if ($scope.widget.inputs) {
                    $scope.widget.inputs.forEach(function(input) {
                        if (input.listen === true &&
onInputChange.hasOwnProperty(input.id)) {
                            var eventname =
events['webgui.dataExchange'] + '.' + (input.name || input.id);
```

```
                            $scope.$on(eventname, function() {
                                onInputChange[input.id](input);
                            });
                        }
                    });
                }
            };
            listenToInputs();


            // functions called when an output value needs to be
populated (key = output.id)
            // Should be extended as needed
            var onExportOutput = {
                WIDGET_SELECTIONS: function(output) {
                    var wsOutput =
xxxMyWidgetService.buildWidgetSelections($scope.widget); // adapt this
code with your own parameters
                    if (wsOutput) {
                        dataExchangeService.setData(output.name,
wsOutput);
                    }
                }
            };


            // function called each time widget outputs need to be
updated
            var exportOutputs = function() {
                if($scope.widget.outputs) {
                    $scope.widget.outputs.forEach(function(output) {
                        onExportOutput[output.id](output);
                    });
                }
            };
            exportOutputs(); // init


            /**
             * Here are some standard UOC events you can listen to in
order to add some processing at various points of the widget lifecycle
             * Event handlers are bound to the scope of the widget's
controller
             */


            /** Triggered when the widget must be refreshed, for example
when clicking on the button refresh of the widget's toolbar */
```

```
            $scope.$on(events['webgui.widgetRefresh'], function() {
                refresh();
            });

            /** Triggered when the widget refresh must be canceled, for
example when clicking on the button cancel of the widget's toolbar */
            $scope.$on(events['webgui.widgetCancelRefresh'], function()
{
                // stop the spinner
                $scope.widget.busy = false;
            });

            /** Triggered when the size of the view changes, for example
when opening the data selection menu. It is not triggered on window
resize, only on some specific cases */
            $scope.$on(events['webgui.widgetResize'], function() {

            });

            /** Triggered when the configuration of a widget has been
changed using the configuration modal */
            $scope.$on(events['webgui.widgetConfigurationChanged'],
function(/*event, oldConfiguration, newConfiguration*/) {
                redraw();
            });

            /** Triggered when the UOC theme has been changed */
            $scope.$on(events['webgui.themeChanged'], function() {
                themeConfigService.applyThemeCSS('xxx-my-widget'); //
Apply widget specific stylesheet for the selected theme if any
            });

            // Support autoload option (can be set to true in the widget
definition in the view). Usually triggers a refresh. default: true
            if ($scope.widget.autoload !== undefined ?
$scope.widget.autoload : true) {
                // wait for the digest to be over
                // (if this $timeout is removed, the redraw will be
triggered before the widget uniqueId has been populated)
                $timeout(function() {
                    refresh();
                }, 0);
            }
            themeConfigService.applyThemeCSS('xxx-my-widget'); // Apply
widget specific stylesheet for the selected theme if any
```

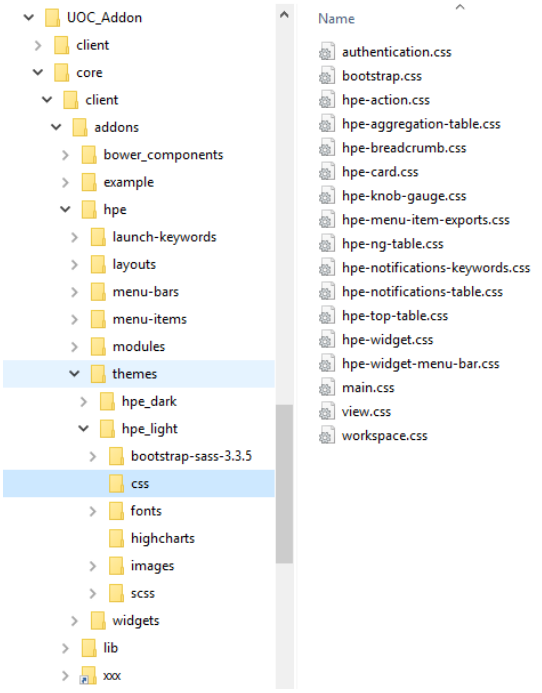
```
        }
    ]).controller('xxxMyWidgetConfigurationController', ['$scope',
        function($scope) {
            // initialization
            if($scope.widgetConfiguration.showTitle === undefined) {
                $scope.widgetConfiguration.showTitle = true;
            }

            $scope.clear = function(object, prop) {
                if(object) {
                    delete object[prop];
                }
            };
        }
    ]);

    return xxxMyWidgetControllers;
});
```

10.12.7 Theme Configuration Support

The Widget leverages an advanced features provided by the add-on theme and the UOC graphical engine while rendering the view. All details of this processing can be read in 13.3 Theme Configuration Support for Components.

IMPORTANT: the SDK generate does not provide an empty skeleton in the UOC themes to prepare any specific customization. It is in charge of the add-on developer to add this file and CSS specific file to support this advanced feature

The widget developer provides a default style for its widget in the widget module and need to integrate the Theme Configuration Services and init it correctly in its controller to support a dynamic customization driven by the add-on theme (done by other integrators, external teams, customers, etc.)



By default, HPE dark and light themes integrates many customization on widgets to follow the HPE branding.

There is a strict naming convention to make sure the UOC graphical engine associate the right CSS file to the right component during the rendering step.

Component Identifier must be unique leverage the add-on identifier

<my add-on id>-<my widget id>.css

In

<install dir>/client/addons/<theme id>/themes/css

Example of Theme Configuration Support in a widget

```
define([
      'commons/events/commons-events',
      'components/theme-config/theme-config-services'
],
function(angular) {
var hpeFormControllers = angular.module('hpeFormControllers', [
'commonsEvents', 'themeConfigServices']);
     hpeFormControllers.controller('hpeFormController', ['events',
function(events, themeConfigService) {
      themeConfigService.applyThemeCSS('hpe-form');
…
// We listen a specific event 'themeChanged' to manage the dynamic
change of theme by end user.
       $scope.$on(events['webgui.themeChanged'], function() {
                        themeConfigService.applyThemeCSS('hpe-form');
                    });
```

NOTE: applyThemeCSS will check in the theme CSS directory if a specific customization for this widget has been done for this theme and if it exists, it overrides the default CSS style implemented by the developer.

More details in 18.9 Theme Configuration Services

10.12.8 Widget events

This controller implement by default all needed JavaScript event to manage the behavior of the widget.

The add-on developer can listen and catch these events to customize the processing at various points of the widget lifecycle:

webgui.widgetRefresh : Triggered when the widget must be refreshed, for example when clicking on the button refresh of the widget's toolbar.

- webgui.widgetRefresh
- webgui.widgetCancelRefresh: Triggered when the widget refresh must be canceled, for example when clicking on the button cancel of the widget's toolbar. See 18.1.4 webgui.widgetCancelRefresh

webgui.widgetResize: Triggered when the size of the view changes, for example when opening the data selection menu. It is not triggered on window resize, only on some specific cases.

- webgui. widgetResize

webgui.widgetConfigurationChanged: Triggered when the configuration of a widget has been changed using the configuration dialog.

- webgui.widget.configurationChanged

webgui.themeChanged: Triggered when the UOC theme has been changed.

- webgui.themeChanged. A typical usecase is to propaget some changes on embedded open sources that manage themes in their own way.

These events are defined in commons-events module ("commonsEvents").

```
        var events = {
                // identifier to use : string to exchange at runtime (should be the same)
                'webgui.widgetRefresh': 'webgui.widgetRefresh',
                'webgui.widgetCancelRefresh': 'webgui.widgetCancelRefresh',
                'webgui.widgetResize': 'webgui.widgetResize',
                'webgui.dataExchange': 'webgui.dataExchange',
                'webgui.widgetConfigurationChanged': 'webgui.widgetConfigurationChanged',
                'webgui.themeChanged': 'webgui.themeChanged',
...
        };
```

10.12.9 Autoload feature

By default, a widget is instanciated in the layout, and start to retrieve its associated data with queries (i.e. autoload feature is true). Some widgets may choose to turn off the autoload property to force the end user to manually click on refresh button to start to collect data and process them for display. IT could be interesting in some cases when the data collection and processing is long or impact too much performance at data server level).

There is an important variable named "autoload" that by default, send an event to start an initial refresh and make sure that as soon as the widget is loaded, the widget need to collect data and display them. This is the default behavior is this property in the widget configuration is not run off explicitly.

Example of widget configuration to turn off the autoload feature:

```
    {
      "id": "AggregationTable",
      "type": "hpe-aggregation-table","title": "Aggregation Table",
      "autoload": false,
      "configuration": {}
}
```

The widget developer has to manage this standard option of UOC Widget with this simple piece of code in the controller.

```
…
// Support autoload option (can be set to true in the widget definition
in the view). Usually triggers a refresh. default: true
   if ($scope.widget.autoload !== undefined ? $scope.widget.autoload :
true) {
          // wait for the digest to be over
          // (if this $timeout is removed, the redraw will be triggered
before the widget uniqueId has been populated)
          $timeout(function() {
               refresh();
           }, 0);
       }
…
```

10.12.10 Logs

UOC relies on the angular logs service ($log) for add-ons client and to open the web browser console of your favorite browser, you usually can press F12. See 2.13 Angular Logs

Example of $log usage in UOC

```
trainingWeatherControllers.controller('trainingWeatherController',
['$scope', '$log',
    function($scope, $log) {
      var logger = $log.getInstance('trainingWeatherController');
…
      logger.error('An error occurred while retrieving data from server:
', err);
```

By default, the $log is generated by the SDK code generator and it is only recommended to use it for your own logs.

IMPORTANT: These logs are not persisted or send to the server side for storage. Goal is to avoid issue with performance and avoid useless request between the web browser and the UOC server.

10.12.11 Message and notifications

Widget totally relies on the common client service message-notifier to send visual message to the notification area. The SDK code generator integrates by default this service and you only need to leverage its api for your own need in your widget. Please refer to 18.10 Message Notifier Services.

Example of message-notifier services in the widget training-weather:

```
define([
…
'components/message-notifier/message-notifier-services',
…
], function(angular)
{
     'use strict';
    var trainingWeatherControllers =
angular.module('trainingWeatherControllers',
['messageNotifierServices']);

    trainingWeatherControllers.controller('trainingWeatherController',
['$scope', '$log', 'messageNotifierService',
    function($scope, $log, messageNotifierService) {
…
                  messageNotifierService.error
('TRAINING_WEATHER_WIDGET.MESSAGES.ERRORS.GET_DATA_ERROR');
…
};
```

10.13 Services File

In the following example generated by the Generator of the SDK, a pre-defined service is implemented with its logger. It integrates the Data Access Services and a default implementation to collect data from the plugin (see getData) and prepare data (see formatData).

This service identified by 'xxxMyWidgetService'. It is ready for implementation.

```
define([
    'angular',
    'lodash',
    'components/data-access/data-access-services'
], function(angular, _) {
    'use strict';
    var xxxMyWidgetServices = angular.module('xxxMyWidgetServices',
['dataAccessServices']);

    xxxMyWidgetServices.service('xxxMyWidgetService', ['$q', '$log',
'dataAccessService',
```

```
        function($q, $log, dataAccessService) {

            var logger = $log.getInstance('xxxMyWidgetService');

            /**
             * Format the data received before using them to display
information.
             * A good choice of format can ease the HTML and controller
parts
             *
             * @method     formatData
             * @param      {object}  data    data to prepare
             * @param      {object}  widget  the widget
             * @return     {object}  formatted data
             */
            var formatData = function(data, widget) {
                logger.debug('formatData', data, widget);

                var result = _.get(data, 'data');

                return result;
            };

              /**
               * Get the formatted data from server.
               * Asynchronous method.
               *
               * @method     getData
               * @param      {object}  widget  the widget
               * @return     {object}  formatted data
               */
              this.getData = function(widget) {
                logger.debug('getData', widget);

                // var deferreds = dataAccessService.getObjects(widget);
// use this line if typeSelection
                var deferreds = dataAccessService.getData(widget); //
use this line if factSelection
                var promises = _.pluck(deferreds, 'promise');
                var deferred = $q.defer();

                $q.all(promises).then(function(results) {
```

```
                    // results.length is equal to the total number of
dataSelections
                    if (!results.length) {
                        deferred.resolve(null);
                    } else {
                        try {
                            //
                            // handle the results and build the
formatted data (use formatData function)
                            //
                            var formattedData = [];
                            results.forEach(function(result) {
                                formattedData.push(formatData(result));
                            });

                            deferred.resolve(formattedData);
                        } catch (e) {
                            deferred.reject(e);
                        }
                    }
                }).catch(function(err) {
                    deferred.reject(err);
                });

                return deferred.promise;
            };

            /**
             * Return widget selections following the unified format
             *
             * @method     buildWidgetSelections
             * @param      {object}  widget         the widget
             * @param      {Array}   selectedValues  list of selected
values to be used to generate the filter part
             * @return     {Array}   array of selections (dataSelection
and dataFilters)
             */
            this.buildWidgetSelections = function(widget,
selectedValues) {
                logger.debug('buildWidgetSelections', widget,
selectedValues);
                return [];
            };
```

```
        }
    ]);
    return xxxMyWidgetServices;
});
```

10.14 Icon

By default, the SDK generator does not generate a preview icon. You can manually customize a specific icon and copy it in the following directory following the naming convention:

<install_dir>\client\public\images\widgets\<my add-on id>-<my-widget id>.png

Default format is PNG and default size is 256x256

IMPORTANT: To support theming, it is recommended to make this icon transparent (background color will apply the color to the widget)

10.15 Widgets Lifecycle

Once they are loaded into the view, widgets are in charge of querying the data to the domain server through the specific plugin. Each value pack knows what server is able to answer these requests and the UOC framework will redirect all pending requests to the right plugins.

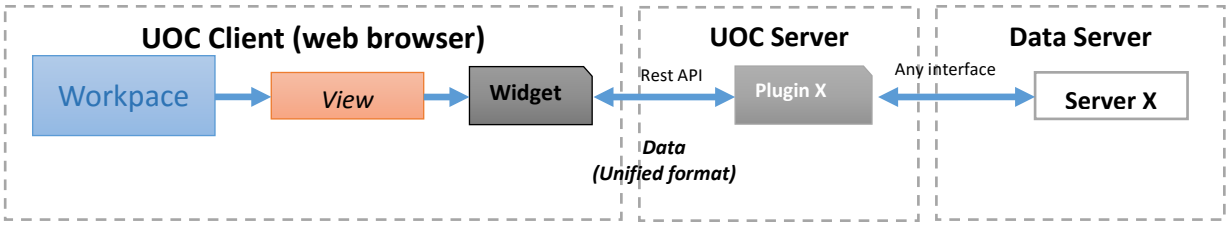


Figure 77 – Widget lifecycle

Interaction between plugin and widget is bi-directional. The widget can collect data and can perform operations.

The widget will only call the plugin to start request to the data server only if the widget has the autoload feature on (default is on). If not, the widget will wait for a manual refresh operation to start this part of the process.

See 10.12.9 Autoload feature to see how to support such behavior from developer point of view.

10.16 Widget Process View

Each widget in charge of his lifecycle based on common services

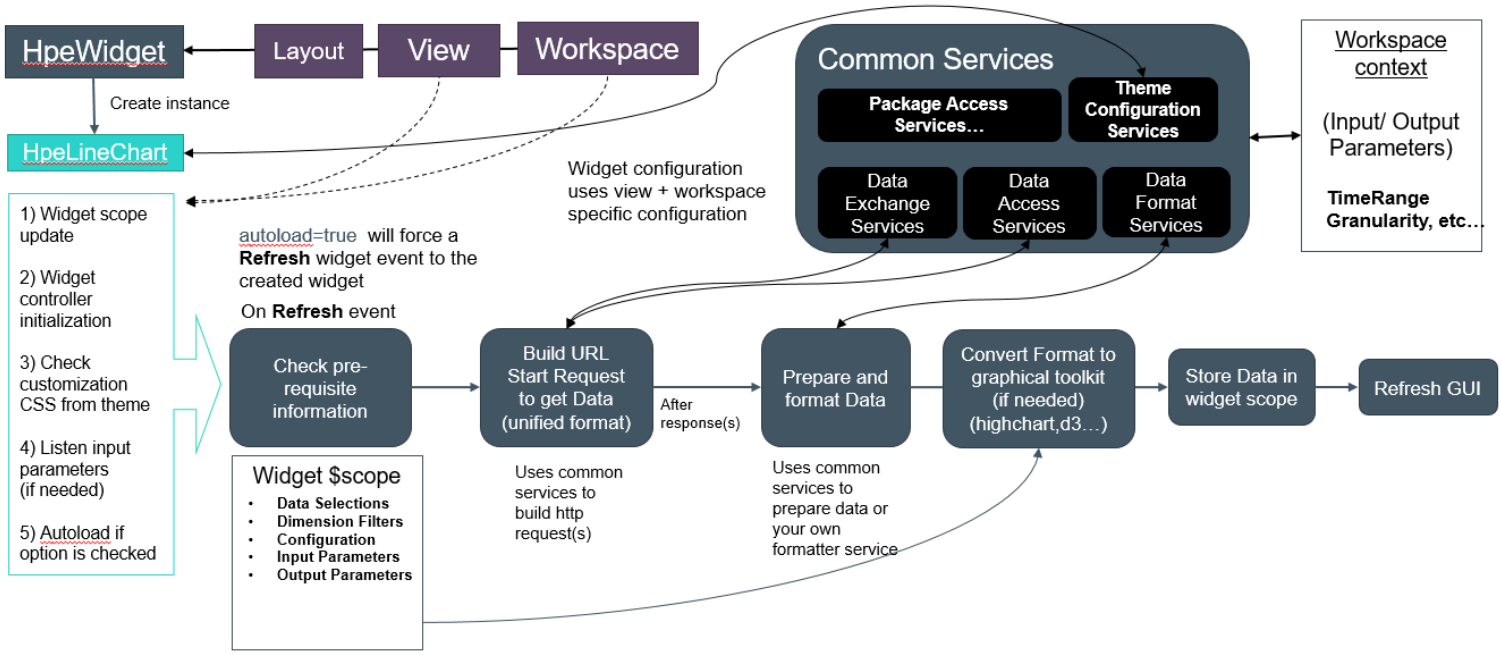


Figure 78 – Add-ons – Widget Process View

10.17 Query Engine

The SDK code generator integrate by default the common client services in charge of performing queries to the plugin to collect data. This service is the Data Access Services. See 18.3 Data Access Services.

It is strongly recommended to implement your data query in your own widget services.

The request relies on $q and promises. See 2.14 Angular Promises for details.

NOTE: A promise represents the result of an asynchronous operation (e.g. a value that is not yet known). A promise is in one of three different states:

- pending - *The initial state of a promise.*
- fulfilled - *The state of a promise representing a successful operation.*
- rejected - *The state of a promise representing a failed operation.*

A deferred represents a chainable utility object with methods to register multiple callbacks into callback queues, invoke callback queues, and relay the success or failure state of any synchronous or asynchronous function.

Read more details about promise and deferred at http://documentup.com/kriskowal/q/

After calling getData() or getObject() methods of the dataExchangeService with your widget JSON configuration, you will get promises and list of deferred. Promises should be handled with $q.all method by passing in either an Array or Object of promises which will call .then() once both are resolved.

Here is a basic implementation to automatically generate request to plugins based on the data selection configuration. This data selection can have one or multiple requests to multiple domain data servers.

```
/**
 * Get the formatted data from server. / Asynchronous method.
 *
 * @method     getData
 * @param      {object}  widget  the widget
 * @return     {object}  formatted data
 */
this.getData = function(widget) {
    logger.debug('getData widget=', widget);


    // var deferreds = dataAccessService.getObjects(widget); // use this line if typeSelection
    var deferreds = dataAccessService.getData(widget); // use this line if factSelection
    var promises = _.pluck(deferreds, 'promise');
    var deferred = $q.defer();


    $q.all(promises).then(function(results) {
        // results.length is equal to the total number of dataSelections
        if (!results.length) {
            deferred.resolve(null);
        } else {
            try {
                // handle the results and build the formatted data (use formatData function)
                var formattedData = [];
                results.forEach(function(result) {
                    formattedData.push(formatData(result));
                });
                deferred.resolve(formattedData);
            } catch (e) {
                deferred.reject(e);
            }
        }
    }).catch(function(err) {
        deferred.reject(err);
    });
    return deferred.promise;
};
```

10.18 Input / Output Parameters

I/O parameters are in charge of the inter communication between components. It provides a lousy dependencies implementing the publish / consume.

The widget controller at initialization needs to listen to all changes on its input parameters with the property 'listen' set to true to receive real-time event and apply a given processing.

Example of widget descriptor describing the Input and output parameters:

```
[{
    "widgetId": "hpe-ng-table",
    "name": "HPE Ng Table",
    "description": "Generic widget table with many options",
    "author": "hpe",
    "version": "1.3",
    "icon": "hpe-ng-table.png",
    "categories": ["HPE"],
    "domains": [],
    "packages": [],
    "templateUrl": "/addons/hpe/widgets/hpe-ng-table/hpe-ng-table.tpl.html",
    "optionsTemplateUrl": "/addons/hpe/widgets/hpe-ng-table/hpe-ng-table.cfg.html",
    "toolbar": {
        "zoom": false,
        "configuration": true,
        "refresh": true,
        "selectors": true
    },
    "inputs": [{
        "id": "TIMERANGE_FROM",
        "description": "Beginning of the selected time period",
        "name": "From",
        "defaultValue": "TODAY"
    }, {
        "id": "TIMERANGE_TO",
        "description": "End of the selected time period",
        "name": "To",
        "defaultValue": "TODAY"
    }, {
        "id": "GRANULARITY",
        "description": "Granularity value in seconds",
        "name": "Granularity",
        "defaultValue": 900
    }, {
        "id": "WIDGET_SELECTIONS",
        "description": "Data selection parameter to use to complete the request"
    }, {
```

```
        "id": "WIDGET_REFRESH",
        "description": "Parameters allowing to refresh the widget on value change"
    }],
    "outputs": [{
        "id": "URL",
        "description": "URL to display inside the Widget IFrame (internal launch)"
    }, {
        "id": "WIDGET_SELECTIONS",
        "description": "Data selection parameter to populate on selection"
    }]
}]
```

10.18.1 Listen Input Parameters changes

Input parameters are defined in the JSON definition

```
"inputs": [{
          "name": "DATEFORM7",
          "id": "TIMERANGE_FROM“
     }, {
           "name": "DATETO7",
            "id": "TIMERANGE_TO",
     }, {
           "name": “myGranularity",
           "id": "GRANULARITY",
           "listen": true
}]
```

Each widget can easily listen this changes adding the following code at initialization of its controller.

```
// Listen for listened input changes
if ($scope.widget.inputs) {
  $scope.widget.inputs.forEach(function(input) {
  if (input.listen) {
     //Trigger refresh on change. Event name is an UOC2 convention
     var eventName = events['webgui.dataExchange'] + '.' + input.name;
     $scope.$on(eventName, function() {
           // this is the function to call when it receives a changes.
           refresh();
      });
  }
  });
}
```

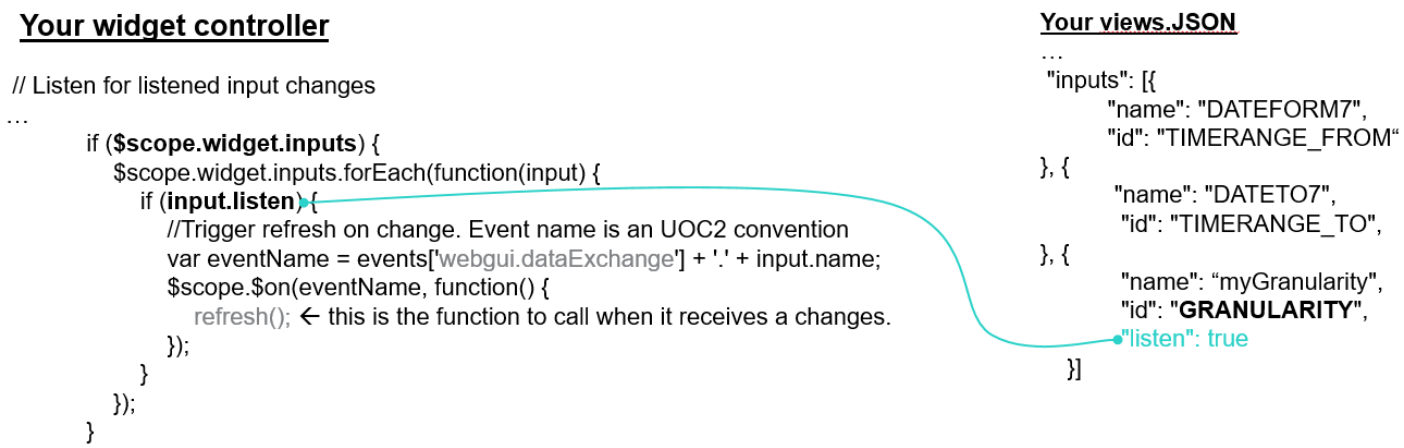


Figure 79 – Add-ons – Widget I/O Parameters management

10.18.2 WIDGET_SELECTIONS parameter

To ease inter communication between heterogeneous widgets, a common format is used WIDGET_SELECTIONS (as an Identifier) is the standard format to exchange data based on the value pack (package) definition between component in Unified Console.

It allows to specify a fine selection of data refining the domain, packages, dimensions and facts –or- object and operations. It also take care of the top filtering.

Most of the HPE UOC widgets supports WIDGET_SELECTIONS as input parameter to communicate with a selection of another widget and output parameter to publish its own selection to others.

It usually defines several parts:

- Data selection
- Data filters
- Top filter

Example of WIDGET_SELECTIONS for OSS Analytics and MBBQoE package

```
[{
    "dataSelection": {
        "factSelection": [{
            "domain": "ossa",
            "package": "MBBQOE_Trial",
            "dimensions": [{
                "id": "BRAND", "folder": ["DEVICE", "Device"]
            }, {
                "id": "TECHNOLOGY", "folder": ["NETWORK", "Cell"]
            }],
            "facts": [{
                "id": "DNS_req_count", "folder": ["DNS", "DNS
Accessibility"]
            }]}]
    },
    "dataFilters": {
        "filters": [{
            "domain": "ossa",
            "package": "MBBQOE_Trial",
            "dimensionFilters": [{
                "dimensionId": "BRAND", "eq": ["Motorola"] }, {
"dimensionId": "TECHNOLOGY","eq": ["3G"] }]
        }]    }
}]
```

This selection defines a query to dimensions Brand / Technology, fact DNS_req_count with a pre-defined filter for Motorola as Brand and 3G for technology.

10.19 Integrate an external Bower Module in your widget

In the UOC runtime kit, several open sources available on the client side are available for your own usage. You can see the complete listing and version in 23.2 Client Open Sources.

It is recommended to leverage the same open sources when possible to ease integration and future evolution. These open sources are regularly upgraded in the UOC kit and tested to grant any regression.

In case you need an additional third party product (open sources or commercial libraries), you will need to define these new dependencies in your add-on.

These dependencies are saved in the file 'bower.json'.

The SDK by default does not define any runtime dependencies and set 2 development dependencies dedicated to support the unit test framework.

```
{
  "name": "hpe-unified-console-dev",
  "description": "Client dependencies for UOC2 add-on development",
  "version": "2.3.0",
  "dependencies": {},
  "devDependencies": {
    "angular-mocks": "1.5.8",
    "chai": "3.5.0"
  },
  "resolutions": {
    "angular": "1.5.8"
}}
```

IMPORTANT: UOC totally relies on the bower package manager to manage dependencies on the front end side. It is strongly recommended to leverage the same tool to ease future migration or version changes, instead of duplicating locally in your project, the JavaScript code without strict tracking.

There are 2 ways to update your bower dependencies:

- Using bower tool command line with the –save flag
- Manually editing the bower.json file in <install dir>/client/

To update your front end dependencies, you need to run the following bower command in your sdk <install dir> directory.

```
bower install
```

You usually download minified version and source version. Detailed documentation : http://bower.io/docs/api

All dependencies will be checked and updated in needed in **<install dir>/client/bower_components**

IMPORTANT: These external dependencies will be part of your distribution in a minified version

Example: Add weather icons in your weather widget. see https://erikflowers.github.io/weather-icons/

1. Add the dependency in your bower.json using the –save flag

```
bower install weather-icons --save
```

NOTE: Bower option -save means to persist this changes in the bower.json. Without this option, you will need to modify manually this dependencies.

You should have a bower.json that look like:

```
{
  "name": "my -addon",
  "description": "Client dependencies for UOC2 add-on development",
  "version": "2.3.0",
  "dependencies": {
          "weather-icons": "2.0.10"
  },
  "devDependencies": {
    "angular-mocks": "1.5.8",
    "chai": "3.5.0"
  },
  "resolutions": {
    "angular": "1.5.8"
}}
```

2. Run bower install to load the dependency in your bower_components directory.

3. Use this new module in your own client code. You will have to also inject correctly services defined by this 3rd party product in your controller following the usual dependency injection of Angular. See 2.6 Dependencies Injection (DI)

Example:

```
define([
    'angular',
    'addons/training/widgets/training-weather/training-weather-
controllers',    // load the widget's controllers
    'css!addons/training/widgets/training-weather/training-weather.css',
// load the CSS stylesheet with require
    'css!addons/training/bower_components/weather-icons/css/weather-
icons.min.css',      // load the CSS stylesheet of weather-icons library
    'css!addons/training/bower_components/weather-icons/css/weather-
icons-wind.min.css'  // load the CSS stylesheet of weather-icons library
],
function(angular) {
… // do not forget to inject your services in angular module
}
```

Chapter 11 Client Add-ons – Menu Bar

11.1 Overview

The Add-on Menu-bar is a main menu for the application. Add-ons HPE provides a default menu bar (hpe-menu-bar) for UOC and to follow the HPE Branding in term of look and feel.

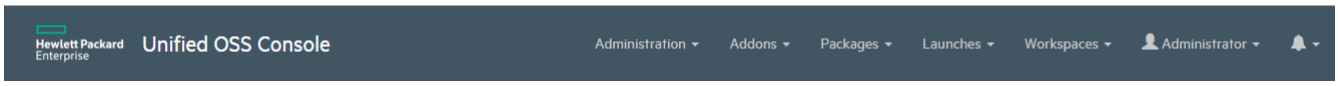


Figure 80 – Add-on HPE Menu Bar (hpe-menu-bar)

It is very likely that your customer ask you to customize a menu-bar, and then you will have to create a new menu-bar to extend the default menu or integrate new menu-items.

UOC gives the full flexibility to customize your own menus mixing add-ons menu-bar and menu-item.

Menu-bar file usually created by a person skilled in web design. It usually require a simple implementation but you need to fully understand technologies like HTML 5, CSS3, Angular Directives, and optionally Twitter Bootstrap.

By default, UOC provides menu-bars in its reusable library and you can extend this graphical library with your own menu-bar.

NOTE: Menu-bars can also integrated in a layout using a dedicated widget (hpe-widget-menu-bar)

All available menu-bars installed on the UOC platform can be browsed through the user interface if you have the permission to browse menu-bar in your role (platform administrator or view designer by default).

Go to the default menu-bar, select Add-ons, then Menu-bars.

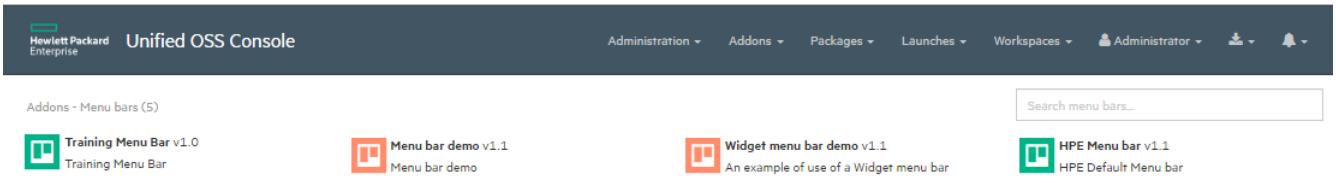


Figure 81 – Add-ons – Menu-bars Management

Menu-bars can be found under <install_dir>/clients/addons/<addonId>/menu-bars

11.2 Structure

To create a new menu-bar, it is strongly recommended to use the SDK generator to generate a working skeleton at the right location based on your Add-on identifier.

Menu-bar is placed under <install_dir>\client\addons\<my add-on Id>\menu-bars\<my add-on id>-<my menu-bar id> directory and will be composed of the following files:

Files	Description
menu-bars.json	Descriptor of the menu-bar Note: The name is fixed and cannot be changed.
<my add-on id>-<my-menu-bar id>.html	Implementation of the menu-bar as HTML file based on bootstrap.
<my add-on id>-<my-menu-bar id>.css	Implementation of the menu-bar style as CSS.

It is recommended to have one directory per menu-bar to ease maintenance and reusability.

11.3 Descriptor

The descriptor is mandatory and defines the menu bar object. Please refer to the object JSON definition in 6.16 Menu Bar

Example of descriptor for menu bar

```
[{
        "menuBarId": "training-menu-bar",
        "name": "Training Menu Bar",
        "description": "Training Menu Bar",
        "author": "hpe",
        "version": "1.0",
        "icon": “training-menu-bar.png",
        "templateUrl": "/addons/training/menu-bars/training-menu-bar/training-menu-bar.html"
}]
```

11.4 Menu Bar File

The menu bar file is a HTML file that describe the HTML tags, CSS style and bootstrap class (responsive, style…). The HTML integrates several menu-item directives. You can easily mixed the default menu-items from UOC and your own custom menu items.

```
<nav class="navbar navbar-default navbar-fixed-top customMenuBar"
role="navigation">


     <style>
          @import url('/addons/training/menu-bars/training-menu-
bar/training-menu-bar.css');
     </style>


     <div class="container-fluid">
          <!-- Brand and toggle get grouped for better mobile display -
->
          <div class="navbar-header">
             <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target=".navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
             </button>


             <hpe-menu-item-logo title="'My Custom Menu'"
link="'http://www.hp.com'"></hpe-menu-item-logo>
          </div>
```

```
            <div class="navbar-collapse collapse">
             <ul class="nav navbar-nav navbar-left">
                 <training-menu-item-operations></training-menu-item-
operations>

                 <has-permission values="browse workspace">
                    <hpe-menu-item-workspaces></hpe-menu-item-workspaces>
                 </has-permission>

                 <hpe-menu-item-preferences></hpe-menu-item-preferences>

               </ul>
            </div>
      </div>
</nav>
```

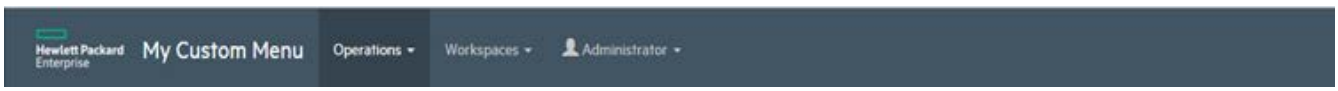
NOTE: You can uses specific directives created by UOC to add security rules in block of HTML.

- <hasPermission> will check if the connected user has the mandatory permission before generating the associated HTML.
- <hasRole> will check if the connected user has the mandatory role before generating the associated HTML.

More details in 16.2 Access Control API.

Example of this custom menu-bar 'training-menu-bar'

This example integrates default hpe-menu-item-logo, a custom menu-item named 'training-menu-item-operations', hpe-



menu-item-workspaces and hpe-menu-item-preferences.

Figure 82 – Example of custom menu-bar

11.5 Icon

By default, the SDK generator does not generate a preview icon. You can manually customize a specific icon and copy it in the following directory following the naming convention:

<install_dir>\client\public\images\menu-bars\<my add-on id>-<my-menu-bar id>.png

Default format is PNG and default size is 48x48.

Chapter 12 Client Add-ons – Menu Item

12.1 Overview

A menu item is a component of a menu bar (sub-menus, search zones, links…). A menu item can only be visible if integrated into a menu bar (i.e. main menu).

By Default HPE provides several menu-items in its add-ons but it is likely you create your own menu items to enhance the menu-bar. You also can mix custom menu item and HPE menu item to create a new menu bar for your customers.

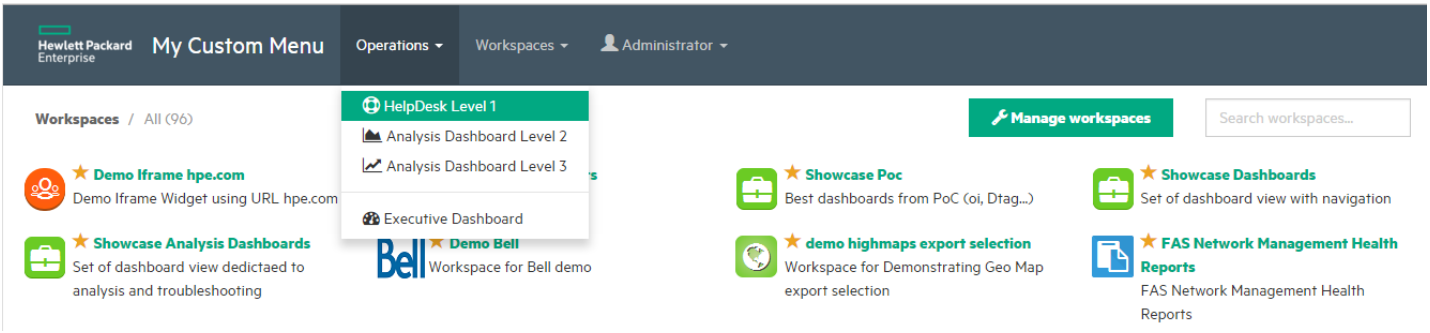


Figure 83 – Example of custom menu-item in a menu-bar

Menu item is an angular module that extend HTML (angular directive).

By default, UOC provides menu-items in its reusable library and you can extend this graphical library with your own menu-item.

All available menu-items installed on the UOC platform can be browsed through the user interface if you have the permission to browse menu-item in your role (platform administrator or view designer by default).

Go to the default menu-item, select Add-ons, then Menu-items.

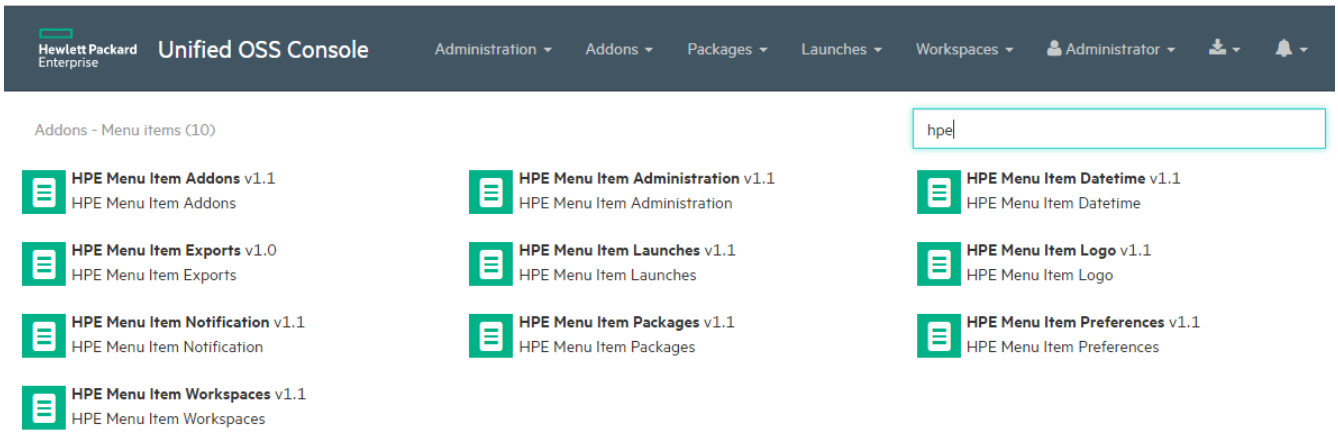
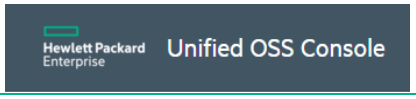
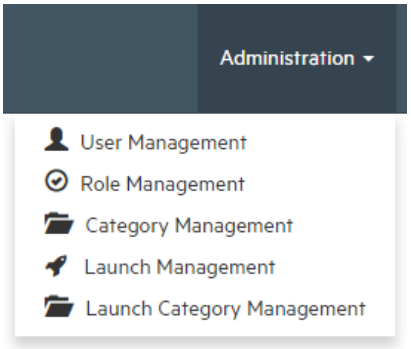
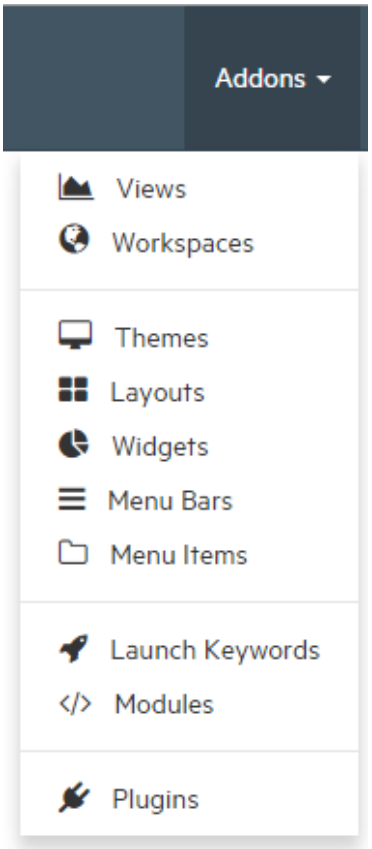


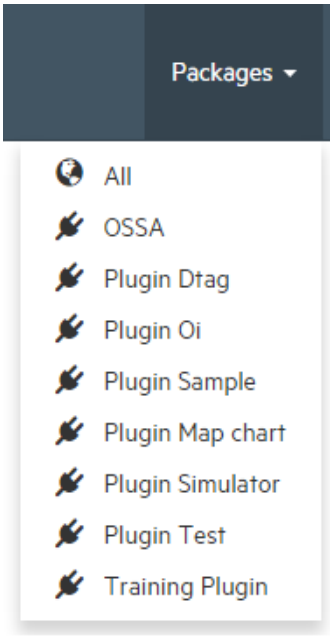
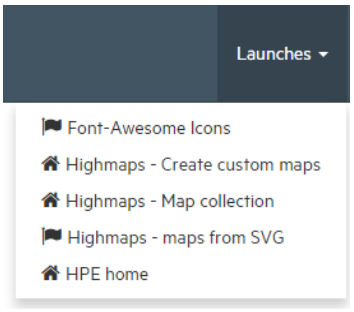
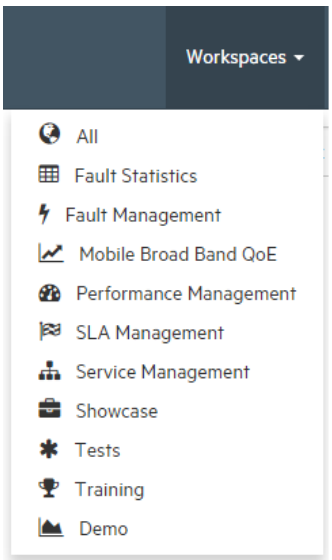
Figure 84 – Add-ons – Menu-items Management

Menu-items can be found under <install_dir>/clients/addons/<addonId>/menu-items

12.2 Default HPE Menu Items

By Default HPE provides several menu-items in its add-ons:

Menu Item / Identifier	Description	Screenshot
HPE Menu Item Logo **(hpe-menu-item-logo)**	Displays a logo, a title and optionally a link to click on	
HPE Menu Item Administration ***(hpe-menu-item-administration)***	Allows a user administrator (user or platform) to access to the platform management screens	
HPE Menu Item Add-ons ***(hpe-menu-item-addons)***	Allows a Platform Administrator to browse available add-ons on the platform and access to the view management and the view designer.	

HPE Menu Item Packages ***(hpe-menu-item-packages)***	Allows a Platform Administrator to browse the available packages available for the platform per domain server.	
HPE Menu Item Launches ***(hpe-menu-item-launches)***	Allows a user to browse and execute global launches available on the platform	
HPE Menu Item Workspaces ***(hpe-menu-item-workspaces)***	Allows a user to access to the workspace management page and browse workspaces, by category, open a workspace and all associated management operations available for the user, depending on his profile and associated permissions.	

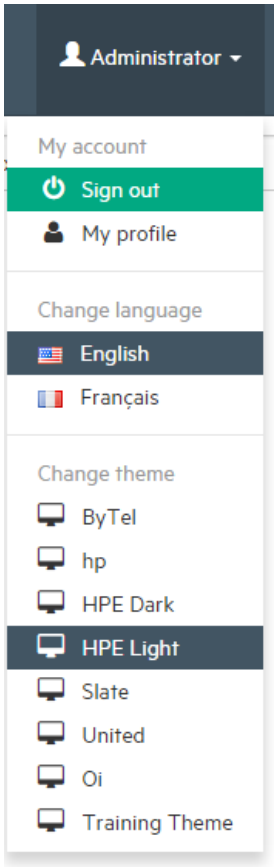
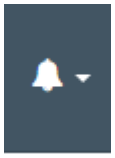
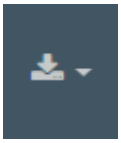
HPE Menu Item Preferences ***(hpe-menu-item-preferences)***	Allows a user to access to the language selection, theme selection, logout button and his personal information.	
HPE Menu Item Notification ***(hpe-menu-item-notification)***	Allows a user to access to the notification console, and see latest messages and their severity level (info, success, error and warning). It also provides a configuration panel to filter these notification messages.	
HPE Menu Item Export ***(hpe-menu-item-exports)***	Allows a user to access to the export file notification feature. It displays a list of exports with their status. (pending, success or error)	
HPE Menu Item Datetime ***(hpe-menu-item-datetime)***	Allows a user to display date and/or time. A time zone and a format can be specified. A title and an icon can be displayed optionally.	 2 examples of HPE Menu Item Datetime

Table 3: HPE Menu Items (Default)

NOTE: HPE Menu Items are under <install_dir>/client/addons/hpe/menu-items

12.3 Structure

Menu-item will be placed under <install_dir>\client\addons\<my add-on Id>\ menu- items\<my add-on id>-<my menu-item id> directory and will be composed of the following files:

Files	Description
menu- items.json	Descriptor of the menu- item Note: The name is fixed and cannot be changed.
<my add-on id>-<my menu- item id>.tpl.html	Definition of the template HTML to describe the HTML new tag to use. Naming convention: postfix this file by –tpl.html
<my add-on id>-<my menu- item id>.html	Implementation of the menu- item as HTML file based on bootstrap.
<my add-on id>-<my menu- item id>.css	Implementation of the menu- item style as CSS.
<my add-on id>-<my menu- item id>.js	Implementation of the main module for the menu item
<my add-on id>-<my menu- item id>-controllers.js	Implementation of controllers for the menu item Naming convention: postfix this file by –controllers.js
<my add-on id>-<my menu- item id>-directives..js	Implementation of directives for the menu item Naming convention: postfix this file by –directives.js
<language code>.json	One localization JSON file per supported language codes. *(e.g. en-us.json, fr-fr.json,…)*

12.4 Descriptor

The descriptor is mandatory and defines the menu item object. Please refer to the object JSON definition in 6.17 Menu Items

Example of descriptor for menu item

```
[{
   "menuItemId": "xxx-my-menu-item",
   "name": "My Menu Item",
   "description": "This is my menu item",
   "author": "xxx",
   "version": "1.0",
   "icon": "",
   "templateUrl": "/addons/xxx/menu-items/xxx-my-menu-item/xxx-my-menu-item.tpl.html"
}]
```

12.5 Directives File

Please refer to 2.8 Angular Directives for detailed explanation about directives.

This file describe exactly how to render the custom tag using standard HTML (templateURL).

```
define([
    'angular',
```

```
    'addons/xxx/menu-items/xxx-my-menu-item/xxx-my-menu-item-
controllers',
    'css!addons/xxx/menu-items/xxx-my-menu-item/xxx-my-menu-item.css'
], function(angular) {
    'use strict';
    var xxxMyMenuItemDirectives =
angular.module('xxxMyMenuItemDirectives', ['xxxMyMenuItemControllers']);

    xxxMyMenuItemDirectives.directive('xxxMyMenuItem', [
        function() {
            return {
                restrict: 'E',
                replace: true,
                controller: 'xxxMyMenuItemController',
                scope: { },     // isolated scope
                templateUrl: 'addons/xxx/menu-items/xxx-my-menu-
item/xxx-my-menu-item.html'
            };
        }
    ]);
    return xxxMyMenuItemDirectives;
});
```

12.6 Template File

The template file (*.tpl.html) is a file specific to UOC that in charge of describing the HTML extension available to use the menu item in the menu bar.

```
<xxx-my-menu-item></xxx-my-menu-item>
```

NOTE: It is possible to inject some data in the isolated scope like any angular directives.

Example hpe-menu-item-logo that have parameters for title and link.

<hpe-menu-item-logo title="title" link="link"></hpe-menu-item-logo>

12.7 HTML File

The HTML file implement the HTML rendering to use after replacing the custom tag (directive). This file follow the HTML 5 / CSS 3 standard and usually leverage bootstrap to provide a responsive design to the web page. It also have to take care of the theming support provided by UOC.

By default, the code generator generates a dropdown menu that navigate to a pre-defined workspace and support localization. it integrate support for angular translate through the directive **<translate>**

```
<li class="dropdown" data-dropdown>
    <a data-ng-href="" class="dropdown-toggle" data-dropdown-toggle>
        <translate>XXX_MY_MENU_ITEM.OPERATIONS</translate> <b
class="caret"></b>
    </a>
    <ul class="dropdown-menu">
        <li>
            <a data-ng-href="/workspaces/value_pack_sample_workspace1">
                <i class="fa fa-support"></i>
                <translate>XXX_MY_MENU_ITEM.OPEN_WKS</translate>
            </a>
        </li>
    </ul>
</li>
```

12.8 CSS File

The CSS file implement the style associated to the HTML classes. This is important to isolate the CSS of this module, to avoid breaking the display of other parts of the product. It is a common mistake to customize style that have a global impact. So, it is mandatory to follow this syntax:

```
.xxx-my-menu-item {
…my CSS style here for my menu item
}
```

12.9 Controllers File

The controllers file is very important file, it is in charge of the interaction between the model (data) and the rendering part (view). Controller is basically a JavaScript constructor function that is used to augment the Angular Scope. When the directive is loaded, angular instantiate a new controller. A new **child scope** will be created and made available as an injectable parameter to the Controller's constructor function as `$scope`.

Please refer to the official angular documentation to get details on scope : https://docs.angularjs.org/guide/scope

By default, a controller and its logger are available for implementation . The service in charge of the theme management and common events are also automatically generated.

Please read the associated details in 18.9 Theme Configuration Services

```
define([
    'angular',
    'lodash',
    'commons/events/commons-events',
    'components/theme-config/theme-config-services'
], function(angular, _) {
    'use strict';
    var xxxMyMenuItemControllers =
angular.module('xxxMyMenuItemControllers', ['commonsEvents',
'themeConfigServices']);


    xxxMyMenuItemControllers.controller('xxxMyMenuItemController',
['$scope', '$log', 'events', 'themeConfigService',
        function($scope, $log, events, themeConfigService) {
            var logger = $log.getInstance('xxxMyMenuItemController');


            // event theme changed
            $scope.$on(events['webgui.themeChanged'], function() {
                themeConfigService.applyThemeCSS('xxx-my-menu-item');
            });
            // Apply menu-item specific stylesheet for the selected
theme if any
```

```
            themeConfigService.applyThemeCSS('xxx-my-menu-item');
        }
    ]);
    return xxxMyMenuItemControllers;
});
```

12.10 Icon

By default, the SDK generator does not generate a preview icon. You can manually customize a specific icon and copy it in the following directory following the naming convention:

<install_dir>\client\public\images\menu-items\<my add-on id>-<my-menu-item id>.png

Default format is PNG and default size is 48x48.

Chapter 13 Client Add-ons – Theme

13.1 Overview

A Theme is an object in charge of the look and feel of the whole application. It basically contains a customization of Bootstrap (CSS framework) and allows an end user or platform administrator to select a specific user interface rendering.

Themes allow to customize the color of all bootstrap elements (navigation bar, lists, buttons, text, ...) and it is recommended to use a visual editor to create a new theme and refer to the official documentation (see http://getbootstrap.com/customize/).

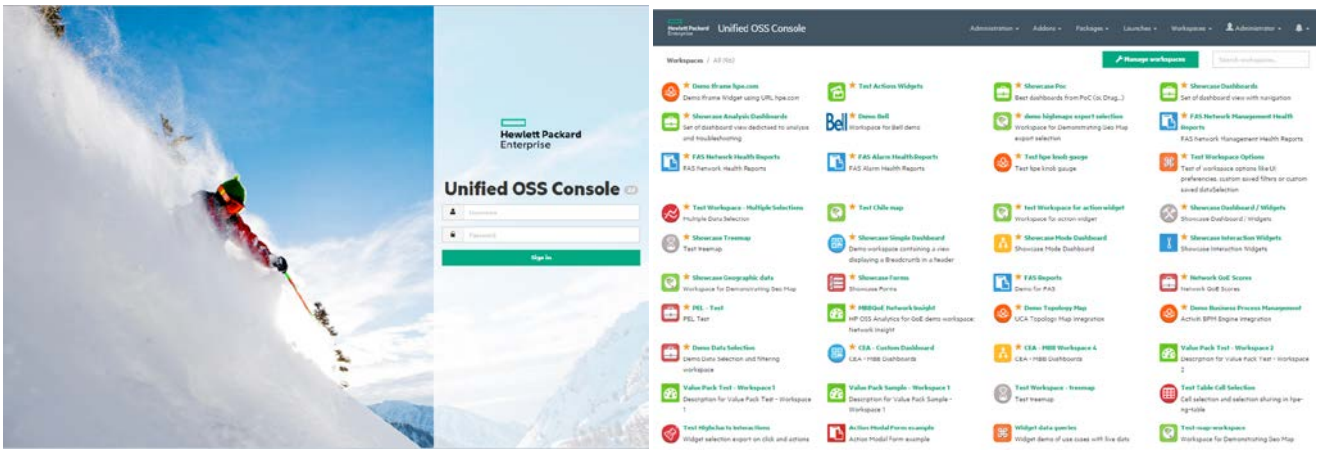


Figure 85 – Theme – HPE (Default)

Add-ons Theme allows the customization of:

- Wallapper
- Login Icon
- Logo icon (main menu)
- Bootstrap Theme
- Highcharts Theme (widet charts)
- All customization of add-ons (menu, widgets, module...)

NOTE: Theme can use set for the platform using the platform preference configuration file. It is also possible then to change the title, the associated URL link and the displayed version.

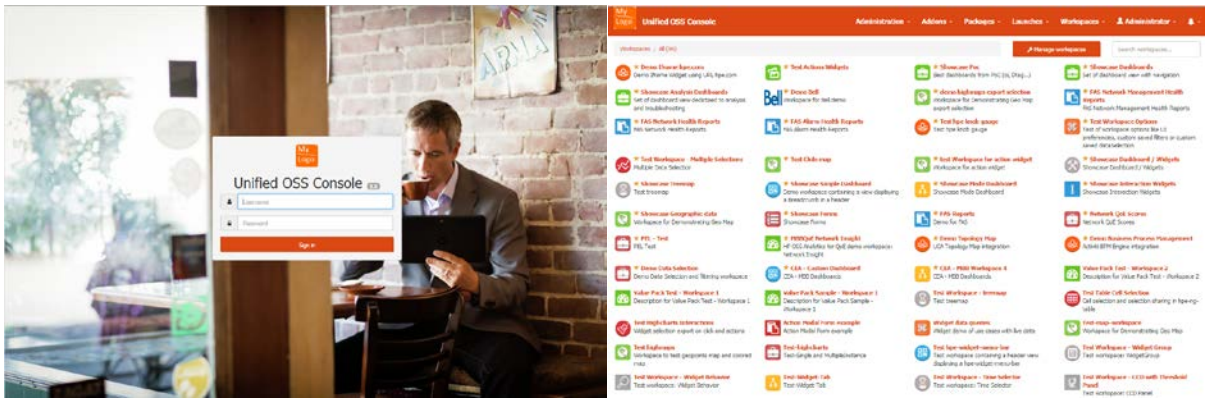


Figure 86 – Example of custom Theme

An Add-ons developer with web design skills can easily extend the list of themes available for the UOC platform. It is strongly recommended to use the SDK Theme generator to build a working skeleton ready for customization (see 5.10 How to generate an add-on client Theme) .

NOTE: Even if Bootstrap is originally developed using LESS, UOC is based on the official Sass port

It is also possible to refine the theme used by charts widget customizing the highcharts json file.

Figure 87 – Theme enhancement – chart widgets support theming'

NOTE: Highcharts Theme is under <install_dir>/client/addons/hpe/themes/<themeId>/highcharts.json

13.2 Default HPE Themes

The following themes are installed by default as Client Add-ons:

- HPE Light (default theme)
- HPE Dark

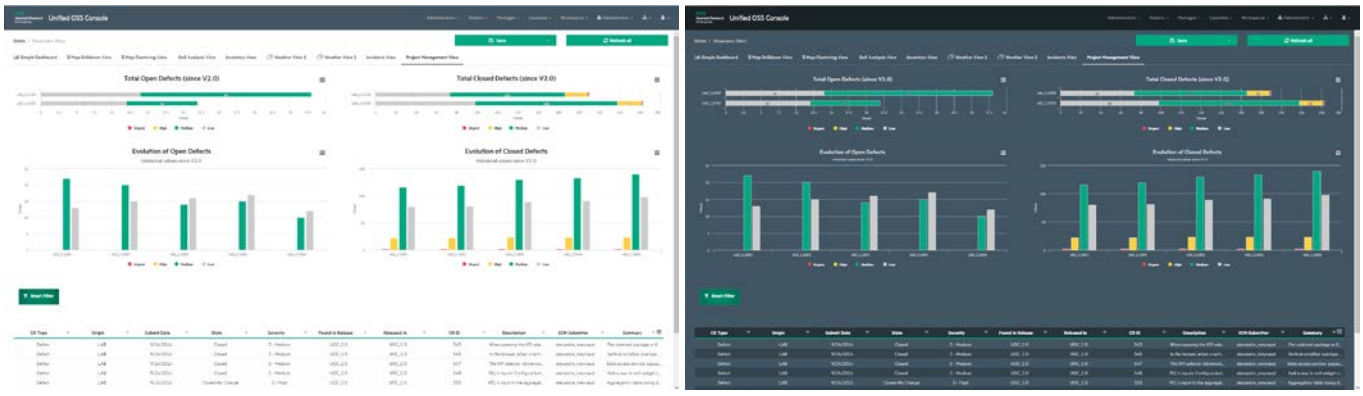


Figure 88 - HPE Light and dark themes

NOTE: HPE Theme is under <install_dir>/client/addons/hpe/themes

As you can see in the examples, every single graphical components can be styled according to the theme, in order to create a coherent and aesthetic feeling, improving the UI/UX along improving the brand visual impact.

13.3 Theme Configuration Support for Components

UOC provides an advanced mechanism to override default style defined by existing components like HPE widgets, menus… A dedicated client services is available to be integrated in your own component to leverage automatically this features. See 18.9 Theme Configuration Services.

Basically, the UOC Graphical engine will apply in order, CSS Theme customization then check if there is a specific customization done for the widget. If yes, the engine will override changes for the final rendering.

The processing is summarized in the following figure:

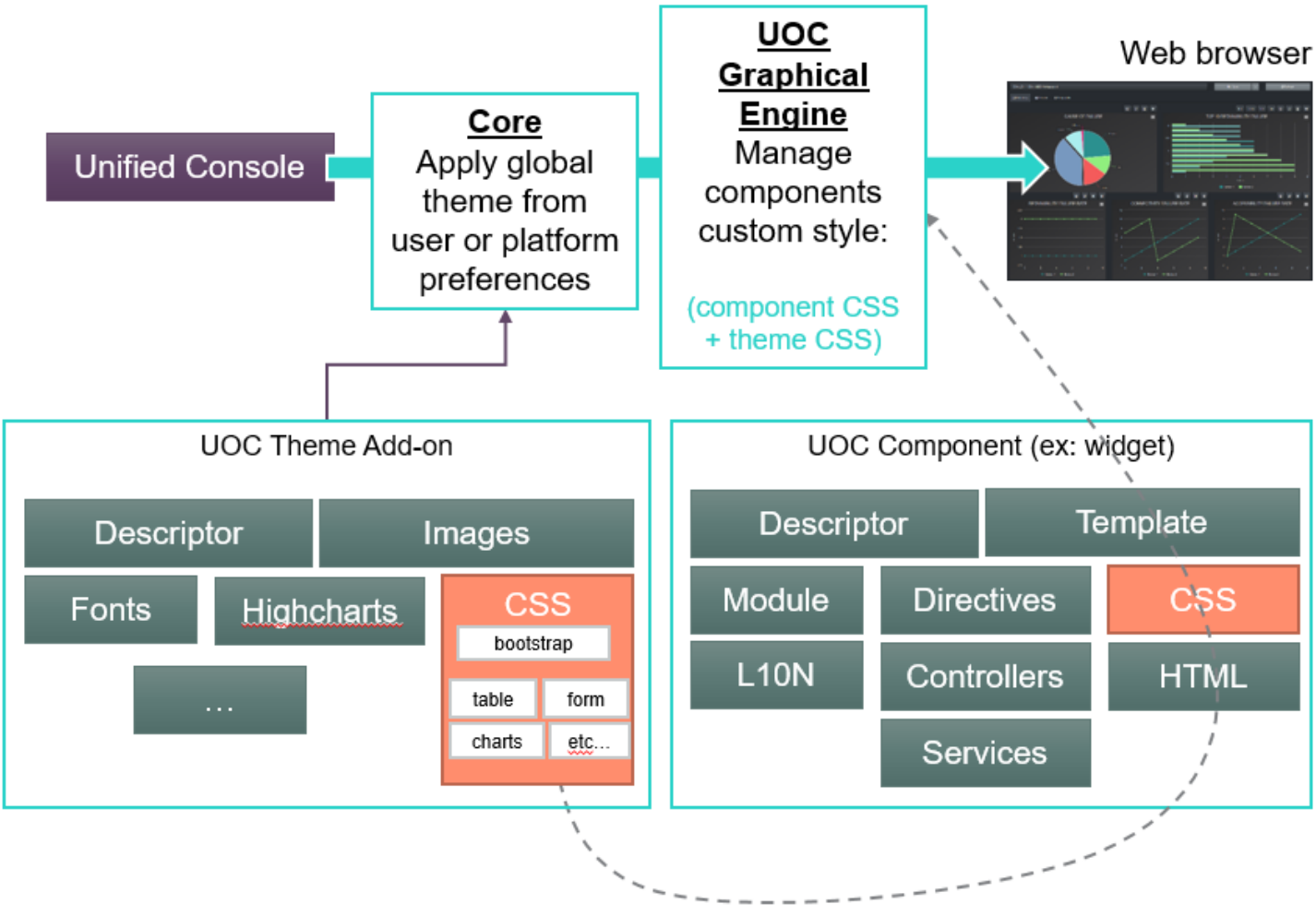


Figure 89 – Theme Configuration Support with components and custom styles.

Example:

<id of the component>.css (ex: hpe-breadcrump.css) in the CSS directory of the UOC theme indicate to the UOC graphical engine to override theme style by the component style. This CSS should only integrate changes compare to the default implementation.

13.4 Syntactically Awesome Stylesheets (SASS)

Sass is a CSS pre-processor. It means that Sass will compile your .scss files into .css ones. Check the following part about compilation.

There are two syntaxes available for Sass. The first, known as SCSS (Sassy CSS), is an extension of the CSS syntax. This means that every valid CSS stylesheet is a valid SCSS file with the same meaning. In addition, SCSS understands most CSS hacks and vendor-specific syntax, such as IE's old filter syntax. This syntax is enhanced with the Sass features such as variables, functions, mixins, etc. Files using this syntax have the .scss extension.
The second and older syntax, known as the indented syntax (or sometimes just "Sass"), provides a more concise way of writing CSS.

UOC is using the SCSS syntax and leverage most of Sass main features for its themes.
The SDK theme generator simply sets some variables and compile a new theme for you.

More details about Sass can be found here : http://sass-lang.com/

13.5 Structure

Theme will be placed under <install_dir>\client\addons\<my add-on Id>\themes\<my add-on id>-<my theme id> directory and will be composed of the following files:

Files	Description
themes.json	Descriptor of the theme
compile-sass.bat	Script for Windows to compile Sass files of the theme and generate CSS files
compile-sass.sh	Script for Linux to compile Sass files of the theme and generate CSS files
css	After compilation of the theme, this directory is generated with CSS files used at runtime.
scss	Sass CSS directory
• components	Directory where each UOC component can optionally define a specific customization to override the style implemented by default. *Naming: <addon-id>-<component id>.scss*
• _hpe-variables.scss	Color variables extracted from the HPE brand. *Note: This file is included in other files and never compiled directly.*
• _bootstrap-variables.scss	Bootstrap variables. Uncomment and replace to customize the default bootstrap theme. *Note: This file is included in other files and never compiled directly.*
• <my add-on id>-<my theme id>.scss	Bootstrap customization not only possible using variables. It overrides bootstrap selectors. *Note: This file contains only modifications of Bootstrap.*
• main.scss	UOC specific customizations that are not part of Bootstrap (helper classes, fixes, etc.)
images	Directory to store specific images for the theme
• branding	Directory to store images to customize the branding of the theme.
o favicon.ico	Favicon to display in the web browser associated to the URL of the application (must be 16 x 16)
o <theme id>.png	Icon of the theme to use in add-ons management. (must be 48 x 48)

o login.png	Icon to use for the login page (local authentication only). Identity Provider provides the login page for SAML authentication.
o logo.png	Icon to use as logo in the main menu bar, next to the title. (recommended size: width x 136)
• wallpapers	Directory to store wallpapers of the theme
o wallpaper.png	Active wallpaper for the theme used for the login page (local authentication only)
highcharts	Directory specific for Highcharts product.
• highcharts.json	Specific theme part for Highcharts graphical library. (JSON format) Check these links for more details https://github.com/highcharts/highcharts/tree/master/js/themes http://www.highcharts.com/docs/chart-design-and-style/themes
fonts	Directory to store font families of the theme. By default you will have HPE Simple and MetricHPE (font dedicated to HPE product).
bootstrap-sass-3.3.x	Directory to store official Sass part of bootstrap.

Here the detailed directories:

```
    compile-sass.bat          Script for theme compiling on windows
    compile-sass.sh           Script for theme compiling on unix
    compileComponents.txt     Example of a watch command on all theme components
    compileMain.txt           Example of a watch command on main.scss
    compileTheme.txt          Example of a watch command on bootstrap.scss
    themes.json               Descriptor

├──bootstrap-sass-3.3.5       Original bootstrap files. No modifications should be
    │                         made in this folder. Extracted and trimmed from
    └──assets                 https://github.com/twbs/bootstrap-sass
        ├──fonts
        │
        └──stylesheets
            │   _bootstrap.scss     Imported in <my-theme>.scss
            │
            └──bootstrap
                │   _*.scss        Bootstrap components (if you need to check how
                │                  it is implemented.
                └──mixins
                    _*.scss       Bootstrap mixins. If you need to check its
                                  implementation or to re-use them

├──css                            All the.scss files will be compiled into .css
    *.css                         and their .map in this folder.
    *.css.map                     Files are compiled, you should not modify them
                                  directly.

├──fonts                      Custom fonts files


├──highcharts                 Highcharts configuration file (colors, fonts, …)
    highcharts.json

├──images
```

```
    ├──branding
    │       favicon.ico       Favicon (can be seen in tabs)
    │       <theme-id>.png    Icon used to represent the theme in the plaform
    │       login.png         Logo used in the login screen
    │       logo.png          Logo used in the navbar
    │
    └──wallpapers
            wallpaper.png     Background image used in the login screen

└──scss
    │   _hpe-variables.scss          Color variables from HPE brand chart
    │   _bootstrap-variables.scss    Variables to customize bootstrap
    │   <theme-id>.scss              Bootstrap-specific customization
    │   main.scss                    UOC-specific customization
    │
    └──components
            *.scss                   Files that will compile into .css files used
                                     by the themeConfigService
```

13.6 Descriptor

The descriptor is mandatory and defines the theme object. Please refer to the object JSON definition in 6.18 Themes

Example of descriptor for a theme

```
[{
        "themeId": "hpe_light",
        "name": "HPE Light",
        "description": "HP Enterprise Theme (Light)",
        "author": "hpe",
        "version": "1.0",
        "icon": "hpe_light.png",
        "location": "/addons/hpe/themes/hpe_light"
}]
```

13.7 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. https://tools.ietf.org/html/rfc2318

After compilation of the theme (scss) , this directory is generated with CSS files used at runtime.

13.8 <my add-on id>-<my theme id>.scss

This file imports :

- <theme_dir>/scss/_hpe-variables.scss that define hpe colors
- <theme_dir>/scss/_bootstrap-variables.scss that overrides default bootstrap variables
- <theme_dir>/bootstrap-sass-3.3.5/assets/stylesheets/bootstrap.scss

This will result in a regular customized bootstrap theme. Any bootstrap theme will work as long as you add the non-standard secondary type of button.

If you want to customize things that are not possible by only changing _bootstrap-variables.scss, you need to:

1. Check out in <theme_dir>/bootstrap-sass-3.3.5/assets/stylesheets/bootstrap/ how the bootstrap component you want to customize is implemented
2. Use the same selectors to override styles the cleanest way
3. Override values in <my-theme>.scss

By default, HPE and generated themes comes with several bootstrap customization such as :

- Buttons (new type, and style)
- Breadcrumb
- Nav (tabs, pills, stacked, ...)
- Pagination
- Forms
- Tooltips and Popovers

This file will be compiled as <theme_dir>/css/bootstrap.css and <theme_dir>/css/bootstrap.css.map using the following command :

```
node-sass scss\<theme-id>.scss css\bootstrap.css --source-map true
```

13.9 _hpe-variables.scss

This file sets color variables coming from HPE color palette according to HPE Brand guidelines. They should be mostly used in bootstrap-variables, and should be avoided in other files.

NOTE: This file start with an underscore "_" meaning that this file won't be compiled.

13.10 _bootstrap-variables.scss

The original file can be found at : https://github.com/twbs/bootstrap-sass/tree/v3.3.5/templates/project.

This file is all the variables default values used in bootstrap. Changing these variables allows a lot of customization of the bootstrap theme. To do so, you need to uncomment the corresponding line and set a value.

Since this file come from an another repository, this is not recommended to add custom variables here to ease maintenance when upgrading to the next version.

To facilitate things, we're using as much as possible colors from _hpe-variables.scss.

NOTE: This file start with an underscore "_" meaning that this file won't be compiled.

13.11 main.scss

This file imports :

- <theme_dir>/scss/_hpe-variables.scss that define hpe colors
- <theme_dir>/scss/_bootstrap-variables.scss that overrides default bootstrap variables

This file contain customization for Unified OSS Console, not for bootstrap.

UOC is using it for :

- Loading custom fonts
- Add helpers (background color, text color, border color, ...)

- Custom animations (action-spin)
- Global layout declaration
- CSS Icon generation

This file will be compiled as <theme_dir>/css/main.css and <theme_dir>/css/main.css.map using the following command :

```
node-sass -w scss\main.scss css\main.css --source-map true
```

13.12 Components

Sometimes, UOC components need theme specific styles, and to achieve this, to access to theme variables.

Those .scss files should be located at <theme_dir>/scss/components and have the name of your component.

These files will be compiled as <theme_dir>/css/<component>.css and <theme_dir>/css/<component>.css.map using the following command :

```
node-sass -w --recursive scss\components\ --output css\ --source-map true
```

The .css compiled files will be used by the 18.9 Theme Configuration Services, especially with the 18.9.2.4 Apply Theme CSS feature.

13.13 bootstrap-sass-3.3.x

This folder is just a copy from https://github.com/twbs/bootstrap-sass/tree/v3.3.5/assets.

images/ and javascripts/ folders have been removed because they had no use for a theme.

No modifications should be made in this folder to ease maintenance.

This files are only useful to check how certain things are implemented so one can override them in a clean way.

13.14 Compilation

To compile new themes, you need to install node-sass as stated in 3.2.5 Install Node Sass.

```
node-sass -w scss\main.scss css\main.css --source-map true
```

node-sass	Call the compiler from Node wrapper
-w	Watch the first argument file(s). The command will hang, and trigger a compilation everytime you save one of the watched file. **It doesn't trigger a compilation when you run the command** Useful for developing purposes (compile automatically at every changes)
scss\main.scss	The SCSS sourcefile
css\main.css	The CSS output
--source-map true	Generate the .css.map along the .css file

To compile components, we use the recursive option from node-sass

```
node-sass -w --recursive scss\components\ --output css\ --source-map true
```

node-sass	Call the compiler from Node wrapper
-w	Watch the first argument file(s). The command will hang, and trigger a compilation everytime you save one of the watched file. **It doesn't trigger a compilation when you run the command** Useful for developing purposes (compile automatically at every changes)
--recursive	Will look for every .scss files recursively in the following folder
scss\components\	Folder for the .scss files
--output css\	CSS output folder
--source-map true	Generate the .css.map along the .css file

In order to compile the full theme, you need to run the three following commands :

node-sass scss\<theme-id>.scss css\bootstrap.css --source-map true	Will compile <theme-id>.scss to generate a bootstrap theme
node-sass scss\main.scss css\main.css --source-map true	Will compile main.scss to generate the UOC-specific styles
node-sass --recursive scss\components\ --output css\ --source-map true	Will compile all the components theme-specific styles

To avoid running these three commands in a row by hand, you can use the following scripts (available in the theme folder) to compile the theme once (no watch argument):

- compile-sass.bat for windows
- compile-sass.sh for unix

NOTE: The commands are relative to the path of the current directory. Run it in the corresponding theme folder.

13.15 Troubleshooting

Debugging .scss files should not be harder than .css files. That's why using the `--source-map true` argument during the compilation from .scss to .css is important.

Generated .css.map files will tell your browser to map .scss lines with the compiled .css that have been interpreted.

As you can see in the next picture, as long as you have .css.map, debugging .scss is just like debugging .css.

Figure 90 : .scss debugging in browser developer tools

13.16 Icon

By default, the SDK generator generate a preview icon with the HPE logo. You can manually customize a specific icon and replace the default one in the following directory, following the naming convention:

<install_dir>\client\addons\<my-addon-id>\themes\<my-theme>\images\branding\<my-theme>.png

Default format is PNG and default size is 48x48.

Chapter 14 Client Add-ons – Module

14.1 Overview

Add-ons Modules are a small piece of code used by component to implement an advanced behavior and allow a high flexibility in customization. It could be a column formatter for table, a custom action to replace a standard behavior, a popup dialog when clicking on an item, a filter to format information, or any angular component (service, controller, directive, filter,...). These modules provides a high level of customization for advanced and specific usage.

The Add-on developer can refer to each widget documentation to know the list of possible customization based on this add-on.

NOTE: All modules are under: **<install_dir>/client/addons/<add-on id>/modules**

There are several type of modules:

- Generic: this is basically a angular module loaded by UOC and available for advanced customization. It allows to create directive that is used by several widget to provide a way to customize a part of the widget (popup dialog, card tile...). It is possible to implement an angular directive, controller, service, filter, ... By default, UOC installs a custom leaflet popup dialog and a custom card tile.
- Filter (or formatter): this module allows some widgets to customize the graphical rendering applying a specific filter or formatter. This is basically an angular filter module used by the widget. By default, UOC install a filter sample to format column.

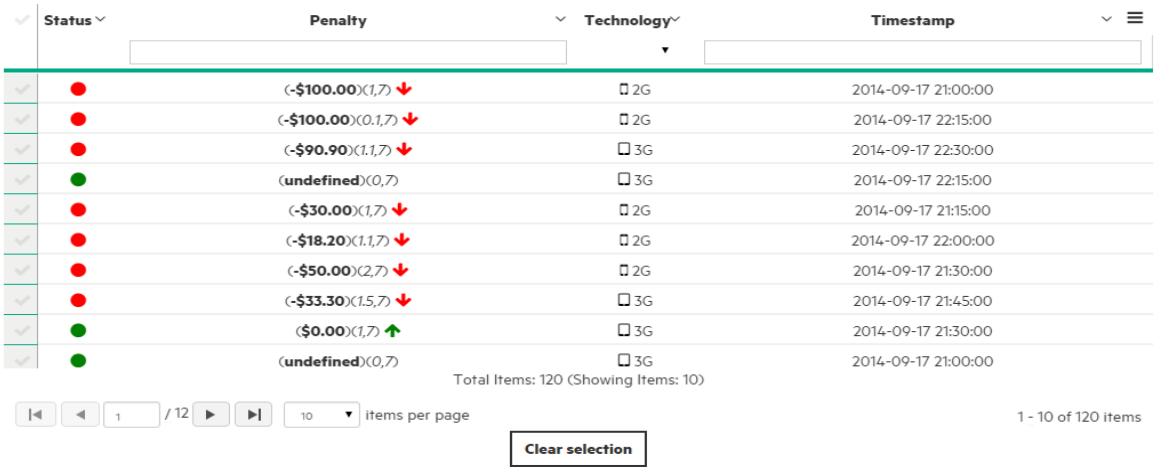


Figure 91: Example of custom formatter in a hpe-ng-table

- Action: This module allow to implement a custom action executed by the UOC Action Service. By default UOC provides a set of pre-defined action (navigation to URL, navigation to view id, set output parameter, execute a launch, and call a custom module). This is basically a angular service implementation used by UOC.
- Form Control: This module extends the graphical control used in forms by widgets: hpe-form, hpe-search or action to popup form and execute operations. These modules are extension of angular schema form (see http://schemaform.io for details and available extensions written by the open source community). By default, UOC install support for date picker, toggle, and simple text.

By default, UOC provides modules in its reusable library and you can extend this library with your own module.

All available modules installed on the UOC platform can be browsed through the user interface if you have the permission to browse module in your role (platform administrator or view designer by default).

Go to the default menu-bar, select Add-ons, then Modules.

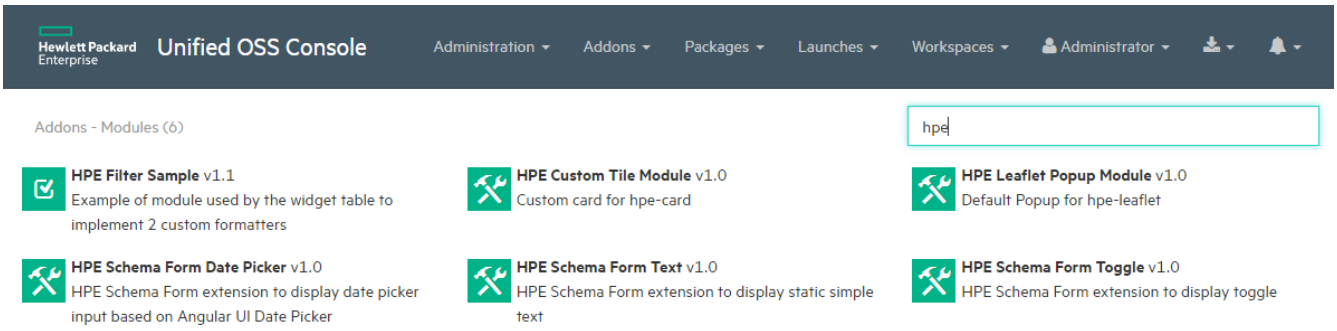


Figure 92 – Add-ons – Modules Management

14.2 Icon

By default, the SDK generator does not generate a preview icon. You can manually customize a specific icon and copy it in the following directory following the naming convention:

<install_dir>\client\public\images\modules\<my add-on id>-<my-module id>.png

Default format is PNG and default size is 48x48.

By default, several icons are installed in <install_dir>\client\public\images\modules\

Basic icons are: default.png, filter.png, code.png, service.png and formatter.png that exist in many colors (light gray, dark gray, green, orange, slate, purple, turquoise)

14.3 Descriptor

Descriptor is a mandatory file because it is used by UOC to identify and load the module at startup. It also contains all information associated to this module (presentation name, version, author, etc.)

Please refer to the object JSON definition in 6.19 Modules

Module descriptor is under <install_dir>\client\addons\<my add-on Id>\modules\<my add-on id>-<my module id>\modules.json

Example of descriptor for a module action (addon ProjectMgt) that define a custom action for conversion.

```
[{
        "id": "convert-dataselection-module",
        "name": "Convert DataSelection module",
        "type": "action",
        "description": "Allow to convert DataSeleciton between 2 files and map fact values and dimension values",
        "author": "hpe",
        "version": "1.0",
        "icon": "code.png",
        "location": "/addons/ProjectMgt/modules/convert-dataselection-module"
}]
```

14.4 Module Type Generic

The module generic is able to integrate any kind of component of Angular (directive, controller, filter, service…). This custom code can be used for many advanced usages.

14.4.1 Structure

To create a new module generic, it is strongly recommended to use the SDK generator to generate a working skeleton at the right location based on your Add-on identifier.

Module will be place under <install_dir>\client\addons\<my add-on Id>\modules\<my add-on id>-<my module id> directory and will be composed of the following files:

Files	Description
modules.json	Descriptor of the module
<my add-on id>-<my-module id>.tpl.html	Definition of the template HTML to describe the HTML new tag to use. Naming convention: postfix this file by –tpl.html
<my add-on id>-<my-module id>.html	Implementation of the module as HTML file
<my add-on id>-<my module id>.css	Implementation of the module style as CSS.
<my add-on id>-<my-module id>.js	Implementation of the main module for the module
<my add-on id>-<my-module id>-controllers.js	Implementation of controllers for the module Naming convention: postfix this file by –controllers.js
<my add-on id>-<my-module id>-directives.js	Implementation of directives for the module Naming convention: postfix this file by –directives.js
<my add-on id>-<my-module id>-services.js	Implementation of services for the module Naming convention: postfix this file by –services.js
<my add-on id>-<my-module id>-filters.js	Implementation of filters for the module Naming convention: postfix this file by –filters.js
<language code>.json	One localization JSON file per supported language codes. Naming convention: <language code>.json *(e.g. en-us.json, fr-fr.json,…)*

It is recommended to have one directory per module to ease maintenance and reusability.

14.4.2 Module File

The main module is automatically loaded by UOC at startup, and is dedicated defines other dependencies to load too. By default a module generic loads angular directives file (if any) and angular filters file (if any). The generator also integrate support of localization automatically.

NOTE:. See https://github.com/angular-translate/angular-translate for details of the localization management in angular module

```
'use strict';
define([
    'angular',
    'components/language-config/language-config',
```

```
    'addons/xxx/modules/xxx-my-module/xxx-my-module-directives',
    'addons/xxx/modules/xxx-my-module/xxx-my-module-filters'
], function(angular) {
    var xxxMyModule = angular.module('xxxMyModule',
['pascalprecht.translate', 'xxxMyModuleDirectives',
'xxxMyModuleFilters']);


     xxxMyModule.run(['$translatePartialLoader',
          function($translatePartialLoader) {
             $translatePartialLoader.addPart('/addons/xxx/modules/xxx-
my-module');
          }
     ]);
    return xxxMyModule;
});
```

14.4.3 Template File

The template file (*.tpl.html) is a file specific to UOC that in charge of describing the HTML extension available to use the module.

```
<xxx-my-module></xxx-my-module>
```

14.4.4 Directives File

NOTE:. This file is only generated if you answer Yes during the generation with the code generator.

If you selected to generate directives file, then the generator will also generate the associated HTML and CSS files. These file describe exactly how to render the custom tag using standard HTML (templateURL).

```
define([
    'angular',
    'addons/xxx/modules/xxx-my-module/xxx-my-module-controllers'
], function(angular) {
    'use strict';
    var xxxMyModuleDirectives = angular.module('xxxMyModuleDirectives',
['xxxMyModuleControllers']);


    xxxMyModuleDirectives.directive('xxxMyModule', [
        function() {
             return {
                restrict: 'E',
                replace: true,
                scope: {
```

```
                },
                controller: 'xxxMyModuleController',
                templateUrl: 'addons/xxx/modules/xxx-my-module/xxx-my-
module.html'
            };
        }
    ]);
    return xxxMyModuleDirectives;
});
```

The directive generated can be used with the following tag in HTML file. UOC usually invokes directive as element like:

```
<xxx-my-module></xxx-my-module>
```

IMPORTANT: You need to use the kebab case format in HTML i.e. All lowercase with - separating words.

14.4.5 HTML File

NOTE:. This file is only generated if you answer Yes to generate directives during the generation with the code generator.

The HTML file implement the HTML rendering to use after replacing the custom tag (directive). This file follow the HTML 5 / CSS 3 standard and usually leverage bootstrap tp provide a responsive design to the web page. It also have to take care of the theiming support provided by UOC.

By default, it integrate support for angular translate through the directive **<translate>**

```
<div class="xxx-my-module">
     <!-- xxx-my-module Code here -->
     <p class="text-center"><translate>XXX_MY_MODULE.TODO</translate>
xxx-my-module</p>
</div>
```

14.4.6 CSS File

NOTE:. This file is only generated if you answer Yes to generate directives during the generation with the code generator.

The CSS file implement the style associated to the HTML classes. This is important to isolate the CSS of this module, to avoid breaking the display of other parts of the product. It is a common mistake to customize style that have a global impact. So, it is mandatory to follow this syntax:

```
.xxx-my-module {
…my CSS style here for my module
}
```

14.4.7 Controllers File

NOTE:. This file is only generated if you answer Yes during the generation with the code generator.

By default, a controller and its logger are available for implementation . The services associated to the module are also injected in the controller and ready to be used.

```
define([
    'angular',
    'addons/xxx/modules/xxx-my-module/xxx-my-module-services'
], function(angular) {
    'use strict';
    var xxxMyModuleControllers =
angular.module('xxxMyModuleControllers', ['xxxMyModuleServices']);


    xxxMyModuleControllers.controller('xxxMyModuleController',
['$scope', '$log',
        function($scope, $log) {
            // client-side enriched angular logger
            var logger = $log.getInstance('xxxMyModuleController');
        }
    ]);


    return xxxMyModuleControllers;
});
```

14.4.8 Services File

NOTE:. This file is only generated if you answer Yes during the generation with the code generator.

In the following example generated by the Generator of the SDK, a pre-defined service is implemented with its logger. This service identified by "xxxMyModuleService' and do not implement any specific behavior. It is ready for implementation.

```
define([
    'angular'
], function(angular) {
    'use strict';
    var xxxMyModuleServices = angular.module('xxxMyModuleServices', []);


    xxxMyModuleServices.service('xxxMyModuleService', ['$log',
        function($log) {
```

```
            var logger = $log.getInstance('xxxMyModuleService');
        }
    ]);
    return xxxMyModuleServices;
});
```

14.4.9 Filters File

NOTE:. This file is only generated if you answer Yes during the generation with the code generator.

In the following example generated by the Generator of the SDK, a pre-defined function is implemented with its logger.

This filter identified by 'xxxMyModuleFilter' does not implement any filtering feature. It just return the input value without transformation, and log with debug level the input value received.

```
define([
    'angular'
], function(angular) {
    'use strict';
    var xxxMyModuleFilters = angular.module('xxxMyModuleFilters', []);

    xxxMyModuleFilters.filter('xxxMyModuleFilter', ['$log',
        function($log) {
            var logger = $log.getInstance('xxxMyModuleFilter');
            return function(input) {
               logger.debug('xxxMyModuleFilter input=' + input);
               return input;
            };

        }
    ]);

    return xxxMyModuleFilters;
});
```

You can use this filter in JavaScript, HTML.

Examples

This filter can be used in HTML or JavaScript.

```
{{ 'Hello world !' | xxxMyModuleFilter }}
```

And also it can be used in JavaScript.

```
$filter('xxxMyModuleFilter')( 'Hello world !')
```

14.4.10 Test Files

NOTE:. These files are only generated if you answer Yes during the generation with the code generator.

Generator is able to generate automatic tests file skeleton for services, directives and controllers. It is recommended to enhance these test file to grant a good test coverage each build of the product. By default, test files checked that all dependencies between these angular components are correctly done and will not raise at runtime any issue in case of different loading order.

14.5 Module Type Filter

14.5.1 Structure

To create a new module filter, it is strongly recommended to use the SDK generator to generate a working skeleton at the right location based on your Add-on identifier.

Module will be place under <install_dir>\client\addons\<my add-on Id>\modules\<my add-on id>-<my module id> directory and will be composed of the following files:

Files	Description
modules.json	Descriptor of the module
<my add-on id>-<my- module id>.js	Implementation of the main module for the module
<my add-on id>-<my- module id>-filters.js	Implementation of filters for the module. Naming convention: postfix this file by –filters.js

It is recommended to have one directory per module to ease maintenance and reusability.

14.5.2 Module File

The main module is automatically loaded by UOC at startup, and is dedicated defines other dependencies to load too. By default a module filter loads angular filters file to make sure this filter (based on an unique identifier) is available.

NOTE: Main module can also setup the localization leveraging angular translate mechanism. See https://github.com/angular-translate/angular-translate for details.

```
'use strict';


define([
    'angular',
    'addons/xxx/modules/xxx-my-module/xxx-my-module-filters'
], function(angular) {
    var xxxMyModule = angular.module('xxxMyModule',
['xxxMyModuleFilters']);


    return xxxMyModule;
});
```

14.5.3 Filters File

The filters file is in charge of implementing one or multiple filters for this module. Filter or formatter in UOC is based on Angular Filter component. Please refer to the official documentation for more details:
https://docs.angularjs.org/api/ng/filter/filter and https://docs.angularjs.org/guide/filter

In the following example generated by the Generator of the SDK, a pre-defined function is implemented with its logger.

This filter identified by 'xxxMyModuleFilter' implements the following behavior:

- logs in debug level the input value (raw data to filter)
- add a HTML bold tag
- logs in debug level the output value (filtered data)
- return the filter value

```
define([
    'angular'
], function(angular) {
    'use strict';
    var xxxMyModuleFilters = angular.module('xxxMyModuleFilters', []);

    xxxMyModuleFilters.filter('xxxMyModuleFilter', ['$log',
        function($log) {
              var logger = $log.getInstance('xxxMyModuleFilter');
              return function(input) {
                 logger.debug('xxxMyModuleFilter input=' + input);
                 var output = '(<b>' + input + '</b>)';
                 logger.debug('xxxMyModuleFilter output=', output);
                 return output;
             };
        }
    ]);

    return xxxMyModuleFilters;
});
```

You can use this filter in JavaScript, HTML and render in bold style any value.

Examples

This filter can be used in HTML or JavaScript.

```
{{ 'Hello world !' | xxxMyModuleFilter }}
```

And also it can be used in JavaScript.

```
$filter('xxxMyModuleFilter')( 'Hello world !')
```

NOTE: Filter defined in module can be used in hpe-ng-table to format a column.

14.6 Module Type Action

14.6.1 Structure

To create a new module action, it is strongly recommended to use the SDK generator to generate a working skeleton at the right location based on your Add-on identifier.

Module will be place under <install_dir>\client\addons\<my add-on Id>\modules\<my add-on id>-<my module id> directory and will be composed of the following files:

Files	Description
modules.json	Descriptor of the module
<my add-on id>-<my- module id>.js	Implementation of the main module for the module
<my add-on id>-<my- module id>-services.js	Implementation of services for the module Naming convention: postfix this file by –services.js

It is recommended to have one directory per module to ease maintenance and reusability.

NOTE: Do not forget that an angular service is a singleton.

14.6.2 Module File

The main module is automatically loaded by UOC at startup, and is dedicated defines other dependencies to load too. By default a module action loads angular services file to make sure these services (based on an unique identifier) are available.

NOTE: Main module can also setup the localization leveraging angular translate mechanism.

See https://github.com/angular-translate/angular-translate for details.

```
'use strict';
define([
    'angular',
    'addons/xxx/modules/xxx-my-module/xxx-my-module-services'
], function(angular) {
    var xxxMyModule = angular.module('xxxMyModule',
['xxxMyModuleServices']);

    return xxxMyModule;
});
```

14.6.3 Services File

The services file is in charge of implementing one or multiple services for this module. Actions in UOC are based on Angular service component. Please refer to the official documentation for more details: https://docs.angularjs.org/guide/services

In the following example generated by the Generator of the SDK, a pre-defined service is implemented with its logger. This service identified by "xxxMyModuleService' implements the following behavior:

When the action named 'xxxMyModuleService' is run, (e.g. when a user clicks on an item or select a row... of a widget), then a debug log indicate this action will be run with its associated action definition and options. A last log indicate the custom action ended.

In the default implementation, the action leverages the Data Exchange Services (public client services. See 18.2 Data Exchange Services) and publish an output parameter *'ACTION-MODULE'* with the value '*success'.*

Purpose of this example is mainly to describe how to integrate common UOC services in an action module.

```
define([
    'angular',
    'components/data-exchange/data-exchange-services'
], function(angular) {
    'use strict';
    var xxxMyModuleServices = angular.module('xxxMyModuleServices',
['dataExchangeServices']);


    xxxMyModuleServices.service('xxxMyModuleService', ['$log',
'dataExchangeService',
        function($log, dataExchangeService) {
            var logger = $log.getInstance('xxxMyModuleService');
            this.run = function(actionDefinition, options) {
               logger.debug('xxxMyModuleService BEGIN');
               logger.debug('xxxMyModuleService run with
actionDefinition='+actionDefinition);
               logger.debug('xxxMyModuleService run with
options='+options);
               dataExchangeService.setData('ACTION-MODULE', 'success');
               logger.debug('xxxMyModuleService END');
            };
        }
    ]);
    return xxxMyModuleServices;
});
```

Chapter 15 Client Add-ons – Launch Keywords

15.1 Overview

A launch keyword defines a placeholder key in a launch URL definition that will be solved dynamically by the UOC by browsing all launch keywords available. This keyword can also be solved from a widget and then use a context (ex: customer identifier, a selected information, a period of time …)

Launch keywords are organized by group to ease the search of all these keywords available in the library.

There are 2 types of launch keywords:

- Global: Only a widget component can solve them because it focus on a specific syntax and context of execution.
- Contextual: These keywords can be solved anywhere in the application. They do not require a given context. It could be information set in configuration file, I/O parameters.

The launch engine will try to solve all launch global keywords first, and then ask to the widget to solve the contextual launch keywords, those are keywords that the widget is the only thing able to solve it, dynamically. At the end, the entire URL should be built and ready for use.

So, the Add-ons Launch keywords are a small pieces of code used by the UOC launch engine to dynamically resolved some keywords with the current value of the context of the module (e.g. customer identifier selected, the port of a server, identifier of the connected user …)

The Add-on developer can refer to each widget documentation to know the list of possible launch keyword and the 15.2 Default Launch Keywords for the default global keywords.

NOTE: All modules are under: <install_dir>/client/addons/<add-on id>/modules

By default, UOC provides modules in its reusable library and you can extend this library with your own launch keyword resolver module..

All available launch keywords installed on the UOC platform can be browsed through the user interface if you have the permission to browse launch keywords in your role (platform administrator or view designer by default).

Go to the default menu-bar, select Add-ons, then Launch Keywords.

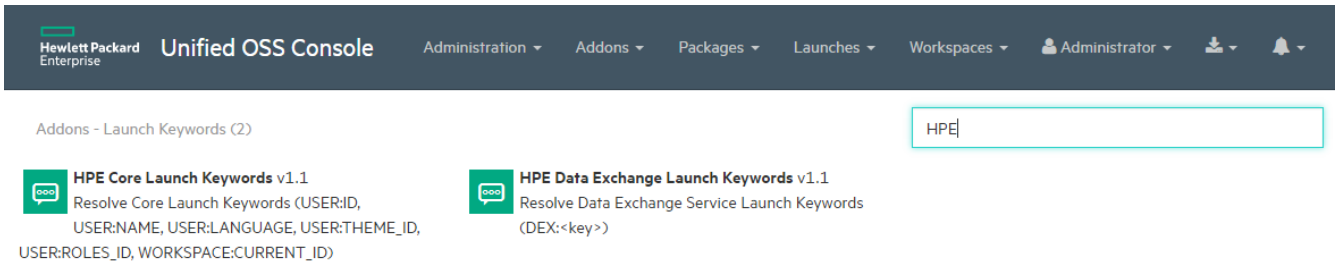


Figure 93 – Add-ons – Launch Keywords Management

15.2 Default Launch Keywords

Several global launch keywords are available by default provided by the core product:

Launch Keywords / Identifier	Description
HPE Core Launch Keyword ***hpe-core-launch-keyword***	Resolve Core Launch Keywords (USER:ID, USER:NAME, USER:LANGUAGE, USER:THEME_ID, USER:ROLES_ID, WORKSPACE:CURRENT_ID)
HPE Data Exchange Launch Keyword ***hpe-data-exchange-launch-keywords***	Resolve Data Exchange Service Launch Keywords (DEX:<key>). Any information stored in the input/Ouput parameters can be used in URL of launches.

15.3 Structure

To create a new launch keyword, it is strongly recommended to use the SDK generator to generate a working skeleton at the right location based on your Add-on identifier.

Launch keyword will be placed under <install_dir>\client\addons\<my add-on Id>\launch-keywords\<my add-on id>-<my launch-keyword id> directory and will be composed of the following files:

Files	Description
launch-keywords.json	Descriptor of the launch keyword
<my add-on id>-<my launch-keyword id>-services.js	Implementation of the launch keyword as an angular services

It is recommended to have one directory per module to ease maintenance and reusability.

15.4 Descriptor

Descriptor is a mandatory file because it is used by UOC to identify and load the launch keyword at startup. It also contains all information associated to this module (presentation name, version, author, etc.)

Please refer to the object JSON definition in 6.20 Launch Keywords

Launch Keywords descriptor is under <install_dir>\client\addons\<my add-on Id>\launch-keywords\<my add-on id>-<my launch-keyword id>\launch-keywords.json

Example of descriptor for a launch keyword (hpe-corelaunch-keywords)

```
{
	"id": "hpe-core-launch-keywords",
	"name": "HPE Core Launch Keywords",
	"description": "Resolve Core Launch Keywords (USER:ID, USER:NAME, USER:LANGUAGE, USER:THEME_ID,
USER:ROLES_ID, WORKSPACE:CURRENT_ID)",
	"author": "hpe",
	"version": "1.1",
	"keywords": [{
		"group": "USER",
		"name": "ID",
		"description": "Return the identifier of the logged user"
	}, {
		"group": "USER",
```

```
                "name": "NAME",
                "description": "Return the display name of the logged user"
        }, {
                "group": "USER",
                "name": "LANGUAGE",
                "description": "Return the language of the logged user"
        }, {
                "group": "USER",
                "name": "THEME_ID",
                "description": "Return the theme identifier of the logged user"
        }, {
                "group": "USER",
                "name": "ROLES_ID",
                "description": "Return the list of roles (id) of the logged user"
        },{
                "group": "WORKSPACE",
                "name": "CURRENT_ID",
                "description": "Return the identifier of the current workspace"
        }
        ]
}
```

15.5 Services File

The services file is in charge of implementing one or multiple launch keyword resolution. Please refer to the official documentation for more details: https://docs.angularjs.org/guide/services

In the following example generated by the Generator of the SDK, a pre-defined service is implemented with its logger that is ready for implementation.

This service identified by "**xxxMyLaunchKeyword'** implements a basic resolution . it generated 2 pre-define groups of launch keywords (GROUP1 and GROUP2) and 2 keywords example per group (KEWORD1 and KEYWORD2)

NOTE: You can define as many group and keywords as you want. There is no limitation.

In the URL, you can enter the following keywords that will be dynamically resolved before executing the URL.

- <GROUP1:KEYWORD1> to return the string 'G1K1',
- <GROUP1:KEYWORD2> to return the string 'G1K2',
- <GROUP2:KEYWORD1> to return the string 'G2K1',
- <GROUP2:KEYWORD2> to return the string 'G2K2',

```
define([
    'angular'
], function(angular) {
    'use strict';
    var xxxMyLaunchKeywordServices =
angular.module('xxxMyLaunchKeywordServices', []);

    xxxMyLaunchKeywordServices.service('xxxMyLaunchKeyword', ['$log',
        function($log) {
```

```
            var logger = $log.getInstance('xxxMyLaunchKeywordService');

            // Group1 keywords
            var getGroup1Keyword = function(name) {
                switch (name) {
                    case 'KEYWORD1':
                        var g1k1 = 'G1K1';
                        logger.debug('getUserKeyword USER:ID=', g1k1);
                        return g1k1;
                    case 'KEYWORD2':
                        var g1k2 = 'G1K2';
                        logger.debug('getUserKeyword USER:NAME=', g1k2);
                        return g1k2;
                }
            };

            // Group2 keywords
            var getGroup2Keyword = function(name) {
                switch (name) {
                    case 'KEYWORD1':
                        var g1k1 = 'G2K1';
                        logger.debug('getUserKeyword USER:ID=', g2k1);
                        return g2k1;
                    case 'KEYWORD2':
                        var g1k2 = 'G2K2';
                        logger.debug('getUserKeyword USER:NAME=', g2k2);
                        return g2k2;
                }
            };

            this.getKeyword = function(group, name) {
                switch (group) {
                    case 'GROUP1':
                        return getGroup1Keyword(name);
                    case 'GROUP2':
                        return getGroup2Keyword(name);
                }
                    logger.debug('getKeyword - group keyword ('+group+') not
found...');
            };
        }
    ]);
```

```
    return xxxMyLaunchKeywordServices;
});
```

15.6 Icon

By default, the SDK generator does not generate a preview icon. You can manually customize a specific icon and copy it in the following directory following the naming convention:

<install_dir>\client\public\images\launch-keywords\<my add-on id>-<my-launch-keyword id>.png

Default format is PNG and default size is 48x48.

Chapter 16 Server Services

UOC Server has several common services available on the server part for integrators or developers of plugins. These common services are Node JS modules that can be injected into your plugins to ease implementation. They are not mandatory to use but ease the integration.

Several common server services are available:

- Common Plugin API
- Access Control API
- Notifications API
- Server Logs API
- Resource Security Logs API
- Session Security Logs API
- Platform Monitoring API
- Platform Statistics API

16.1 Common Plugin API

16.2 Access Control API

16.2.1 Overview

The Server Access Control API is a UOC module in charge of the Role Based Access Control check. This API will aloow or disallow a user to access to a specific URL if the level of protection is not reached.

Access Control provides 4 levels of route protection:

- None: no specific control (no usage of Access Control API)
- Authentication Control : the user is correctly authenticated
- Roles Based Access Control : the user is correctly authenticated and has the correct roles(s)
- Permsissions Based Access Control: the user is correctly authenticated and has the correct permission(s).

16.2.2 Init

The UOC module can be required and used with the following syntax:

```
var accessControl = require(path.join(process.env.ROOT, 'server', 'access-control', 'access-control'));
```

16.2.3 Methods

16.2.3.1 Authentication Control

```
requireAuthRoles()
```

or

```
requireAuthPermissions ()
```

Checks that the user is correctly authenticated but without checking specific roles or specific permissions.

This methods returns error 401 (authentication failure) or 403 (unauthorized user).

16.2.3.2 Authentication and Roles Based Access Control

```
requireAuthRoles(roles, tenant)
```

Check that the user is correctly authenticated with specific roles or tenant.

Parameter	Type	default	Details
roles	Object		Widget definition in JSON containing the data selection, data filter, top filter, I/o parameters...
Tenant *(optional)*	String		Unique Tenant Identifier to validate

This methods returns error 401 (authentication failure) or 403 (unauthorized user).

16.2.3.3 Authentication and Permission Based Access Control

```
requireAuthPermissions(permissions, tenant)
```

Check that the user is correctly authenticated with specific permissions or tenant.

Parameter	Type	default	Details
roles	Object		Widget definition in JSON containing the data selection, data filter, top filter, I/o parameters...
Tenant *(optional)*	String		Unique Tenant Identifier to validate

This methods returns error 401 (authentication failure) or 403 (unauthorized user).

16.2.4 Example

```
var path = require('path');
var packageManager = require('./package-manager');
var accessControl = require(path.join(process.env.ROOT, 'server',
'access-control', 'access-control'));


module.exports = function(app) {
…


// NO SPECIFIC CONTROL
app.get(api + '/packages', packageManager.getPackages);


// AUTHENTICATION CONTROL
app.get(api + '/packages/:packageId/types',
accessControl.requireAuthRoles(), packageManager.getObjectTypes);


// ROLE BASED ACCESS CONTROL. Uses a list of roles ids (Operator_L3)
app.get(api + '/packages/:packageId/types',
accessControl.requireAuthRoles(['Operator_L3']),
packageManager.getObjectTypes);
```

```
// ROLE BASED ACCESS CONTROL. Uses a list of list of permissions ids
(create order or edit order)

app.get(api + '/packages/:packageId/types',
accessControl.requireAuthPermissions([‘create order, edit order']),
packageManager.getObjectTypes);
```

16.3 Notifications API

The publish-subscribe service allows you to propagate notifications through the notification system. It can be used to notify UOC end users

16.3.1 Init

To use the notification system, you need to require the publish-subscribe service:

```
var publishSubscribeService = require(path.join(process.env.ROOT, 'server', 'services', 'publish-subscribe', 'publish-subscribe'));
```

16.3.2 Publish

To publish a notification to subscribers, use the following:

```
publishSubscribeService.publish(compositeKey, data, options, callback);
```

compositeKey : JSON object, referring to a registered notification

Example:

```
{
    id: alertPackage,
    type: alert,
    origin: 'plugin',
    domain: 'plugin_simulator',
    package: 'MBBQOE_Simulation'
}
```

data: JSON object, the data you want to attach to the notification.

Example:

```
{
        "text": {
            "en-us": "Cannot connect to the notification server",
            "fr-fr": "Impossible de se connecter au serveur de notification"
        }
}
```

options: JSON object, possible options you can use to publish a notification

Example:

```
{
“sendToWorkspace”: “workspaceId”
}
```

Supported options properties:

- sendToRoles: Array of string, roles that will be notified
- sendToWorkspace: String, workspace id that will be notified
- sendToView String, view id that will be notified
- keywords Array of string, Update keywords

Properties that should **not** be used (they are used internally by the notification system):

- sendToSessionIds : Array of string, socket ids
- excludeSessionIds: Array of string, socket ids
- broadcast Boolean, Enable broadcast cross server instances

callback: function, will be called upon a notification system error, or once the notification system finished broadcating

Example:

```
var callback = function(err, notification) {
      if (err) {
            logger.error('Cannot publish message:', err);
      }
      else if (notification) {
            logger.info('this notification has been published:',
notification);
      }
};
```

16.3.3 Register

To register one or several notification(s) in the internal register, use the following:

```
publishSubscribeService.register(notifications)
```

Where notifications is an array of notification objects.

16.4 Server Logs API

16.4.1 Overview

The UOC Server has a customizable logger to facilitate troubleshooting of the UOC Platform. This log file supports several levels of log (info, warning, error, warning, debug).

Default level is **warning**. It is also possible to customize the format of the logged line (layout pattern).

NOTE: These logs can be customized by log4JS-node : https://github.com/nomiddlename/log4js-node

The UOC server logs all information related to:

- **Server activities**: General logs related to the server (<install>/logs/server.log)
- **HTTP requests** : Log all http requests done by clients (web browsers). These logs are automatically generated by the Node Express server (<install>/logs/http.log)

Please read chapter Security in the Installation and Configuration Guide to see how to enable/disable logs and how to customize the audit policy and output format.

IMPORTANT: These logs can only be accessed by a platform administrator for troubleshooting reasons and may content sensitive or private information, especially if the debug level is enabled.

16.4.2 Init

Server Logger is available in your plugin to log dedicated information in the server.log file. This logger can be required and used with the following syntax:

```
var path = require('path');

var logger = require(path.join(process.env.ROOT, 'server', 'logger', 'server-logger'));
```

16.4.3 Methods

16.4.3.1 Error log

```
error(msg)
```

As soon as you init your logger object correctly, you have access to the error log.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.2 Error and Console log

```
errorConsole(msg)
```

As soon as you init your logger object correctly, you have access to the error log. This log will be displayed in log file and output console.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.3 Warning log

```
warn(msg)
```

As soon as you init your logger object correctly, you have access to the error log.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.4 Warning and Console log

```
warnConsole(msg)
```

As soon as you init your logger object correctly, you have access to the error log. This log will be displayed in log file and output console.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.5 Information log

```
info(msg)
```

As soon as you init your logger object correctly, you have access to the information log. These log are usually used to provide additional information on demand and are not enabled by default.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.6 Information an Console log

```
infoConsole(msg)
```

As soon as you init your logger object correctly, you have access to the information log. This log will be displayed in log file and output console. These log are usually used to provide additional information on demand and are not enabled by default.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.7 Debug Log

```
debug(msg)
```

As soon as you init your logger object correctly, you have access to the debug log. These log are usually dedicated to the developer to debug or troubleshoot his plugin and are not enabled by default.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.8 Debug and Console Log

```
debugConsole(msg)
```

As soon as you init your logger object correctly, you have access to the debug log. This log will be displayed in log file and output console. These log are usually dedicated to the developer to debug or troubleshoot his plugin and are not enabled by default.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.9 Fatal Log

```
fatal(msg)
```

As soon as you init your logger object correctly, you have access to the fatal log.

Parameter	Type	Details
Msg	String	Text message to log

16.4.3.10 Fatal and Console Log

```
fatalConsole(msg)
```

As soon as you init your logger object correctly, you have access to the fatal log. This log will be displayed in log file and output console.

Parameter	Type	Details
Msg	String	Text message to log

16.4.4 Example

```
var path = require('path');
var logger = require(path.join(process.env.ROOT, 'server', 'logger',
'server-logger'));


logger.debug(‘Information retrieved from the plugin cache. No request to
the data server’);
logger.info(‘User ‘ + user + ‘ connected to the server successfully’);
logger.infoConsole(‘User ‘ + user + ‘ connected to the server
successfully’);
logger.warn(‘Operation has been aborted by the user’);
logger.error(‘This operation failed (error code: ‘ + err + ‘)’);
logger.errorConsole(‘This operation failed (error code: ‘ + err + ‘)’);
 logger.fatal(‘Unexpected error. Server cannot start (error code: ‘ +
err + ‘)’);
```

16.5 Resource Security Logs API

The UOC platform provides several security audit logs. One of them is in charge of tracking the UOC Resource accesses. All resource accesses, operations, applications and data requests. It tracks all actions done by a user on resources are logged with date/time, IP address, user id … to ease the audit (→ <install>/logs/security.log)

Please read chapter Security in the Installation and Configuration Guide to see how to enable/disable logs and how to customize the audit policy and output format.

NOTE: These logs can be customized by log4JS-node : https://github.com/nomiddlename/log4js-node

IMPORTANT: These logs are only accessible to platform administrator and may content sensitive or private information (IP address…)

16.5.1 Init

Security Logger is available in your plugin to log dedicated information in the security.log file. This security logger can be required and used with the following syntax:

```
var path = require('path');

var securityLogger = require(path.join(process.env.ROOT, 'server', 'logger', security-logger'));
```

16.5.2 Methods

16.5.2.1 Workspace Log

```
logWorkspace(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the workspace accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.2 View Log

```
logView(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the view accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.3 Widget Log

```
logWidget(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the widget accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.4 Plugin log

```
logPlugin(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the plugin accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.5 Layout Log

```
logLayout(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the layout accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.6 Category Log

```
logCategory(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the workspace category accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.7 Operation Log

```
logOperation(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the workspace category accesses and operations.

Parameter	Type	Details

Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.8 Launch log

```
logOperation(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the launch accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.9 LaunchCategory Log

```
logLaunchCategory(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the launch category accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.10 Menu Bar log

```
logMenuBar(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the menu bar accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.11 Menu Item log

```
logMenuItem(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the menu item accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.12 Theme Log

```
logTheme(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the theme accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.13 Module Log

```
logModule(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the module accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.14 User Preference Log

```
logUserPreference(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the user preference accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.15 Launch Keyword Log

```
logLaunchKeyword(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the launch keywords accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.16 User Log

```
logUser(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the launch keywords accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.17 Role log

```
logRole(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the role accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.18 Language Log

```
logLanguage(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the language accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.19 Form Log

```
logForm(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the form accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.20 Export Log

```
logExport(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the export accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.21 Permission Log

```
logPermission(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the permission accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.22 Report Log

```
logReport(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the report accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.23 State Log

```
logState(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to the state definition accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.24 Socket Log

```
logSocket(socket, msg)
```

As soon as you init your security logger object correctly, you can log information related to the form accesses and operations.

Parameter	Type	Details
Socket	Object	Socket object. This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.2.25 Publish-Subscribe Log

```
logPublishSubscribe(req, msg)
```

As soon as you init your security logger object correctly, you can log information related to publish-subscribe accesses and operations.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.
Msg	String	Text message to log

16.5.3 Example

```
var path = require('path');

var securityLogger = require(path.join(process.env.ROOT, 'server',
'logger', security-logger'));

securityLogger.logWorkspace(req util.format('Workspace %s has been
accessed', req.params.workspaceId));

securityLogger.logView(req, util.format('View [_id:%s] has been
accessed', req.params.viewId));

securityLogger.logWidget(null, util.format('Browse all widgets
(path:%s)', addonWidgetsPath));

securityLogger.logPlugin(req, util.format('Browse packages of plugin
%s', plugin.id));

securityLogger.logLayout(req, util.format('Layout %s has been accessed',
req.params.layoutId));

securityLogger.logCategory(req, 'Browse all categories defined for
workspaces');

securityLogger.logOperation(req, 'Import operation has been done
successfully');

securityLogger.logLaunch(util.format('Launch Category [_id:%s] has been
created', launchCategory._id));

securityLogger.logLaunchCategory(util.format('Launch Category [_id:%s]
has been updated', launchCategory._id));

securityLogger.logMenuBar(req, util.format('Menu bar %s has been
accessed', req.params.menuBarId));

securityLogger.logMenuItem(req, util.format('Menu item %s has been
accessed', req.params.menuItemId));

securityLogger.logTheme(req, util.format('Theme %s has been accessed',
req.params.themeId));

securityLogger.logModule(req, 'All modules has been accessed');

securityLogger.logUserPreference(req, 'Platform user preferences has
been accessed');

securityLogger.logLaunchKeyword(req, 'Launch Keyword xxx has been
accessed');

securityLogger.logUser(req, util.format('User [user_id:%s] has changed
his preferences', user.id));

securityLogger.logRole(req, util.format('Role [role_id:%s] has been
deleted', body.role_id));

securityLogger.logLanguage(req, 'Languages listing has been accessed');
```

16.6 Session Security Logs API

16.6.1 Overview

The UOC platform provides several security audit logs. One of them is in charge of tracking the UOC **Sessions** to log all information related to the user session (login, logout...). It is possible to see user id, date/time, and IP address to ease the audit (→ <install>/logs/session.log)

Please read chapter Security in the Installation and Configuration Guide to see how to enable/disable logs and how to customize the audit policy and output format.

NOTE: These logs can be customized by log4JS-node : https://github.com/nomiddlename/log4js-node

IMPORTANT: These logs are only accessible to platform administrator and may content sensitive or private information (IP address...)

16.6.2 Init

Session Logger is available in your plugin to log dedicated information in the session.log file. This session logger can be required and used with the following syntax:

```
var path = require('path');

var sessionLogger = require(path.join(process.env.ROOT, 'server', 'logger', session-logger'));
```

16.6.3 Methods

16.6.3.1 Login Log

```
logLogin(req)
```

As soon as you init your session logger object correctly, you can track all user that login to UOC using this API.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.

16.6.3.2 Logout Log

```
logLogout(req)
```

As soon as you init your session logger object correctly, you can track all user that logout to UOC using this API.

Parameter	Type	Details
Req	Object	Http request object, This object will indicate the name / id of the user and optionally his tenant identifier.

16.6.4 Example

```
var path = require('path');
var sessionLogger = require(path.join(process.env.ROOT, 'server',
'logger', session-logger'));


sessionLogger.logLogin(req);
…
sessionLogger.logLogout(req);
```

16.7 Platform Monitoring API

Please contact the R&D Team to get latest official API

Chapter 17 Server REST API

UOC Server has several common services available on the server part for integrators or developers. They supports the REST API protocol.

Common services retrieve the following information installed or defined on the UOC platform:

- Plugins information and their definitions
- Layout
- Widgets
- Roles
- Users information (local authentication mode only)
- Permission
- Workspace categories
- Workspace
- Views
- Platform Monitoring
- Platform Statistics

17.1 Platform Monitoring

17.1.1 Overview

17.1.2 REST API

17.1.2.1 Get Platform Health Status

The UOC server provides a very simple RESTAPI to indicate his health status. This API returns a simple response 'OK' that can be easily be parsed by a load balancer or any external tools to manage the high availability.

Description	HTTP Verb	Rest API
Return the health status	GET	/V1.0/monitoring/server/check

17.2 Plugins

17.2.1 Overview

17.2.2 REST API

17.2.2.1 Get Plugins

Description	
Verb	GET
Path	/V1.0/domains
Description	Get a list of all available domain plugins (active only)
Path Parameters	
Query Parameters	

Response Status Code	200 – OK
Response Object	List of plugins
Errors	

17.2.2.2 Get Packages

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/domains/packages
Description	Get a list of all packages definition for all active plugins
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of packages
Errors	500 : Cannot retrieve the list of packages

17.2.2.3 Get Workspaces

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/domains/workspaces
Description	Get a list of all workspaces for all active plugins
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of workspaces
Errors	500 : Cannot retrieve the list of packages

17.2.2.4 Get Views

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/domains/views
Description	Get a list of all views for all active plugins
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of views
Errors	500 : Cannot retrieve the list of packages

17.3 Layouts

17.3.1 Overview

17.3.2 REST API

17.3.2.1 Get All Layouts

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/layouts
Description	Get a list of all available layouts
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Layouts
Errors	

17.3.2.2 Get Layouts

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/layouts/*<layoutId>*
Description	Get a layout by Identifier
Path Parameters	(*) layoutId is a layout unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Layout
Errors	404: Layout not found

17.4 Widgets

17.4.1 Overview

17.4.2 REST API

17.4.2.1 Get All Widgets

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/widgets
Description	Get a list of all available widgets
Path Parameters	
Query Parameters	

Response Status Code	200 – OK
Response Object	List of Widgets
Errors	

17.4.2.2 Get Widgets

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/widgets/*<widgetId>*
Description	Get a widget by Identifier
Path Parameters	(*) widgetId is a widget unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Widget
Errors	404: Widget not found

17.5 Roles

17.5.1 Overview

17.5.2 REST API

17.5.2.1 Get All Roles

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/roles
Description	Get a list of all available roles
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Roles
Errors	

17.5.2.2 Get All Roles of the User

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/ roles/user
Description	Get a list of all available roles of the connected user
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Roles
Errors	

17.5.2.3 Create Role

This API requires an authenticated token with permission “create role” to be executed.

Description	
Verb	POST
Path	/V1.0/roles
Description	Create a new role
Path Parameters	
Body Parameters	Role
Response Status Code	200 – OK Role [:roleId] has been created
Response Object	
Errors	400: Bad format 403 : Access fordidden 404: Role not found 409: Role already exist 500: Database error

17.5.2.4 Update Role

This API requires an authenticated token with permission “edit role” to be executed.

Description	
Verb	PUT
Path	/V1.0/roles/*<roleId>*
Description	Update an existing role by Identifier
Path Parameters	(*) roleId is a role unique identifier (STRING)
Body Parameters	Roles
Query Parameters	
Response Status Code	200 – OK Role [:roleId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access fordidden 404: Error cannot update the role 500: Database error

17.5.2.5 Delete Role

This API requires an authenticated token with permission “delete role” to be executed.

Description	
Verb	DELETE
Path	/V1.0/roles/*<roleId>*/revision/*:roleRev*
Description	Delete an existing workspace categories by Identifier

Path Parameters	(*) roleId is a role unique identifier (STRING) (*) roleRev is the revision of the role
Query Parameters	
Response Status Code	200 – OK Role [:roleId] [:roleRev] has been deleted
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access fordidden 404: Error cannot delete the role 500: Database error

17.6 Users

17.6.1 Overview

This API is only for managing local user (e.g. stored in the GUI Database CouchDB). It is not related to be used in production. Identity provider used for production will have its own interface to manage users.

17.6.2 REST API

17.6.2.1 Get All Users

This API requires an authenticated token with permission “browse user” to be executed.

Description	
Verb	GET
Path	/V1.0/users
Description	Get a list of all available users
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Users
Errors	

17.6.2.2 Create User

This API requires an authenticated token with permission “create user” to be executed.

Description	
Verb	POST
Path	/V1.0/users
Description	Create a new user
Path Parameters	
Body Parameters	User
Response Status Code	200 – OK User [:userId] has been created
Response Object	

Errors	400: Bad format 403 : Access fordidden 404: User not found 409: User already exist 500: Database error 501: API not supported by SAML authentication mode

17.6.2.3 Update Users

This API requires an authenticated token with permission “edit user” to be executed.

Description	
Verb	PUT
Path	/V1.0/users/*<userId>*
Description	Update an existing user by Identifier
Path Parameters	(*) userId is a user unique identifier (STRING)
Body Parameters	User
Query Parameters	
Response Status Code	200 – OK User [:userId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 :Access fordidden 404: Error cannot update the user 500: Database error

17.6.2.4 Update User Preference

This API requires an authenticated token to be executed.

Description	
Verb	PUT
Path	/V1.0/users/*<userId>*/preferences
Description	Update the user preference of an existing user by Identifier
Path Parameters	(*) userId is a user unique identifier (STRING)
Body Parameters	Preference
Query Parameters	
Response Status Code	200 – OK User [:userId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 :Access fordidden 404: Error cannot update the user

	500: Database error 501: API not supported by SAML authentication mode

17.6.2.5 Update User Password

This API requires an authenticated token to be executed.

Description	
Verb	PUT
Path	/V1.0/users/*<userId>*/password
Description	Update the password of an existing user by Identifier
Path Parameters	(*) userId is a user unique identifier (STRING)
Body Parameters	Old Password (STRING) New Password (STRING)
Query Parameters	
Response Status Code	200 – OK User [:userId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 :Access fordidden 404: Error cannot update the user 500: Database error 501: API not supported by SAML authentication mode

17.6.2.6 Delete User

This API requires an authenticated token with permission “delete user” to be executed.

Description	
Verb	DELETE
Path	/V1.0/roles/*<userId>/revision/:userRev*
Description	Delete an existing user by Identifier
Path Parameters	(*) userId is a user unique identifier (STRING) (*) userRev is the revision of the role
Query Parameters	
Response Status Code	200 – OK User [:userId] [:userRev] has been deleted
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access fordidden 404: Error cannot delete the user 500: Database error

17.7 Permissions

17.7.1 Overview

17.7.2 REST API

17.7.2.1 Get All Permissions

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/permissions
Description	Get a list of all available permissions
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Permissions
Errors	

17.7.2.2 Get All Permissions of the User

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/ permissions / user
Description	Get a list of all available permissions of the connected user
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Permissions
Errors	

17.8 Workspace Categories

17.8.1 Overview

17.8.2 REST API

17.8.2.1 Get All Categories

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/categories
Description	Get a list of all available workspace categories
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Categories

Errors	

17.8.2.2 Get Categories

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/categories/*<categoryId>*
Description	Get a workspace categories by Identifier
Path Parameters	(*) categoryId is a category unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Categories
Errors	404: Category not found 403 : Access forbidden

17.8.2.3 Create Category

This API requires an authenticated token with permission “create category” to be executed.

Description	
Verb	POST
Path	/V1.0/categories
Description	Create a new workspace category
Path Parameters	
Body Parameters	Categories
Response Status Code	200 – OK Category [:categoryId] has been created
Response Object	
Errors	400: Bad format 404: Category not found 403 : Access fordidden 409: Category already exist 500: Database error

17.8.2.4 Update Category

This API requires an authenticated token with permission “edit category” to be executed.

Description	
Verb	PUT
Path	/V1.0/categories/*<categoryId>*
Description	Update an existing workspace categories by Identifier
Path Parameters	(*) categoryId is a categorie unique identifier (STRING)
Body Parameters	Categories
Query Parameters	

Response Status Code	200 – OK Category [:categoriesId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access fordidden 404: Error cannot update the categorie 500: Database error

17.8.2.5 Delete Category

This API requires an authenticated token with permission “delete category” to be executed.

Description	
Verb	DELETE
Path	/V1.0/categories/*<categoryId>*/revision/*:categoryRev*
Description	Delete an existing workspace categories by Identifier
Path Parameters	(*) categoryId is a categorie unique identifier (STRING) (*) categoryRev is the revision of the workspace categorie
Query Parameters	
Response Status Code	200 – OK Category [:categoryId] [:categoryRev] has been deleted
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access fordidden 404: Error cannot delete the categorie 500: Database error

17.9 Workspaces

17.9.1 Overview

17.9.2 REST API

17.9.2.1 Get All Workspaces

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/workspaces
Description	Get a list of all available workspace
Path Parameters	
Query Parameters	
Response Status Code	200 – OK

Response Object	List of Workspaces
Errors	

17.9.2.2 Get All Workspaces (no views details)

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/workspaces/hideViews
Description	Get a list of all available workspace without all the views attributes details.
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Workspaces
Errors	

17.9.2.3 Get Workspaces

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/workspaces /*<workspaceId>*
Description	Get a workspace by Identifier
Path Parameters	(*) workspaceId is a workspace unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Workspace
Errors	403 : Access forbidden 404: Workspace not found

17.9.2.4 Get All Workspaces by categories

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/categories/**<categoryId>**/workspaces
Description	Get a list of all available workspace for a given workspace category
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Workspaces
Errors	

17.9.2.5 Create Workspace

This API requires an authenticated token with permission “create workspace” to be executed.

Description	
Verb	POST
Path	/V1.0/workspaces
Description	Create a new workspace
Path Parameters	
Body Parameters	Workspace
Response Status Code	200 – OK Workspace [:workspaceId] has been created
Response Object	
Errors	400: Bad format 403 : Access forbidden 409: Workspace already exist 500: Database error

17.9.2.6 Update or Create Workspace

This API requires an authenticated token with permission “edit workspace” and “save workspace” to be executed.

Description	
Verb	PUT
Path	/V1.0/workspaces/*<workspaceId>*
Description	Update an existing workspace by Identifier if the workspace already exist else it creates a new workspace
Path Parameters	(*) workspaceId is a workspace unique identifier (STRING)
Body Parameters	Workspace
Query Parameters	
Response Status Code	200 – OK Workspace: [:workspaceId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access forbidden 404: Error cannot update the workspace 500: Database error

17.9.2.7 Update or Create a Private Workspace

This API requires an authenticated token with permission “manage private workspace” to be executed.

Description	
Verb	PUT
Path	/V1.0/workspaces/*<workspaceId>*/private
Description	Update an existing private workspace by Identifier if the workspace already exist else it creates a private workspace

Path Parameters	(*) workspaceId is a workspace unique identifier (STRING)
Body Parameters	Workspace
Query Parameters	
Response Status Code	200 – OK Workspace: [:workspaceId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access forbidden 404: Error cannot update the workspace 500: Database error

17.9.2.8 Delete Workspace

This API requires an authenticated token with permission “delete workspace” to be executed.

Description	
Verb	DELETE
Path	/V1.0/workspaces/*<workspaceId>/revision/:workspaceRev*
Description	Delete an existing workspace by Identifier and a Revision
Path Parameters	(*) workspaceId is a category unique identifier (STRING) (*) workspaceRev is the revision of the workspace (STRING)
Query Parameters	
Response Status Code	200 – OK Workspace [:workspaceId] [:workspaceRev] has been deleted
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access forbidden 404: Error cannot delete the workspace 500: Database error

17.9.2.9 Delete Private Workspace

This API requires an authenticated token with permission “manage private workspace” to be executed.

Description	
Verb	DELETE
Path	/V1.0/workspaces/*<workspaceId>/revision/:workspaceRev*/private
Description	Delete an existing private workspace by Identifier and a Revision. It is like restoring to a public workspace operation.
Path Parameters	(*) workspaceId is a category unique identifier (STRING) (*) workspaceRev is the revision of the workspace (STRING)
Query Parameters	
Response Status Code	200 – OK Workspace [:workspaceId] [:workspaceRev] has been deleted
Response Object	

Errors	400: Bad format or invalid identifier 403 : Access forbidden 404: Error cannot delete the workspace 500: Database error

17.10 Languages

17.10.1 Overview

17.10.2 REST API

17.10.2.1 Get All Languages

Description	
Verb	GET
Path	/V1.0/languages
Description	Get a list of all available languages
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Languages
Errors	

17.10.2.2 Get Language

Description	
Verb	GET
Path	/V1.0/languages /*<languageId>*
Description	Get a language definition by its unique identifier
Path Parameters	(*) languageId is a language unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Language
Errors	403 : Access forbidden 404: Language not found

17.11 Views

17.11.1 Overview

17.11.2 REST API

17.11.2.1 Get All Views

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/views
Description	Get a list of all available workspace
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Views
Errors	

17.11.2.2 Get View

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/ views /*<viewId>*
Description	Get a view by Identifier
Path Parameters	(*) viewId is a view unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	View
Errors	403 : Access fordidden 404: View not found

17.11.2.3 Check if View exist

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/ views /*<viewId>*/exists
Description	Check if a view identifier already exist
Path Parameters	(*) viewId is a view unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	View
Errors	403 : Access fordidden 404: View not found

17.11.2.4 Create View

This API requires an authenticated token with permission “create view” to be executed.

Description	
Verb	POST
Path	/V1.0/views
Description	Create a new view
Path Parameters	
Body Parameters	View
Response Status Code	200 – OK View [:viewId] has been created
Response Object	
Errors	400: Bad format 403 : Access fordidden 409: View already exist 500: Database error

17.11.2.5 Update or Create View

This API requires an authenticated token with permission “edit view” to be executed.

Description	
Verb	PUT
Path	/V1.0/views/*<viewId>*
Description	Update an existing view by Identifier if the view already exist else it creates a new view
Path Parameters	(*) viewId is a view unique identifier (STRING)
Body Parameters	View
Query Parameters	
Response Status Code	200 – OK View: [:viewId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access fordidden 404: Error cannot update the view 500: Database error

17.11.2.6 Delete View

This API requires an authenticated token with permission “delete view” to be executed.

Description	
Verb	DELETE
Path	/V1.0/views/*<viewId>*/revision/*:viewRev*
Description	Delete an existing view by Identifier and a Revision

Path Parameters	(*) viewI=d is a view unique identifier (STRING) (*) viewRev is the revision of the view (STRING)
Query Parameters	
Response Status Code	200 – OK View [:viewId] [:viewRev] has been deleted
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access fordidden 404: Error cannot delete the view 500: Database error

17.12 Launch Categories

17.12.1 Overview

17.12.2 REST API

17.12.2.1 Get All Launch Categories

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/launch-categories
Description	Get a list of all available launch categories
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Launch Categories
Errors	

17.12.2.2 Create Launch Categories

This API requires an authenticated token with permission “create launch category” to be executed.

Description	
Verb	POST
Path	/V1.0/ launch-categories/*<launchCategoryId>*
Description	Create a new launch category
Path Parameters	(*)launchCategoryId is a launch category unique identifier (STRING)
Body Parameters	Launch Category
Response Status Code	200 – OK Launch Category [:categoryId] has been created
Response Object	
Errors	400: Bad format 404: Launch Category not found

	403 : Access forbidden 409: Launch Category already exist 500: Database error

17.12.2.3 Update Launch Categories

This API requires an authenticated token with permission “edit launch category” to be executed.

Description	
Verb	PUT
Path	/V1.0/ launch-categories/*<LaunchCategoryId>*
Description	Update an existing launch categories by Identifier
Path Parameters	(*) launchCategoryId is a launch category unique identifier (STRING)
Body Parameters	Categories
Query Parameters	
Response Status Code	200 – OK Launch Category [:launchCategoriesId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access forbidden 404: Error cannot update the launch category 500: Database error

17.12.2.4 Delete Launch Categories

This API requires an authenticated token with permission “delete launch category” to be executed.

Description	
Verb	DELETE
Path	/V1.0/launch-categories/*<launchCategoryId>/revision/:launchCategoryRev*
Description	Delete an existing launch categories by Identifier
Path Parameters	(*) launchCategoryId is a category unique identifier (STRING) (*) launchCategoryRev is the revision of the launch category
Query Parameters	
Response Status Code	200 – OK Category [:launchCategoryId] [:launchCategoryRev] has been deleted
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access forbidden 404: Error cannot delete the launch category 500: Database error

17.13 Launches

17.13.1 Overview

17.13.2 REST API

17.13.2.1 Get All Launches

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/launches
Description	Get a list of all available launches
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Launches
Errors	

17.13.2.2 Get Launch

This API requires an authenticated token with permission “create launch” to be executed.

Description	
Verb	GET
Path	/V1.0/ launches/*<launchId>*
Description	Get a launches by Identifier
Path Parameters	(*)launchId is a launch unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Launch
Errors	404: Launch not found 403 : Access fordidden

17.13.2.3 Create Launch

This API requires an authenticated token with permission “create launch” to be executed.

Description	
Verb	POST
Path	/V1.0/launches/*<launchId>*
Description	Create a new launch category
Path Parameters	(*)launchId is a launch unique identifier (STRING)
Body Parameters	Launch
Response Status Code	200 – OK Launch [:launchId] has been created
Response Object	

Errors	400: Bad format 404: Launch not found 403 : Access forbidden 409: Launch already exist 500: Database error

17.13.2.4 Update Launches

This API requires an authenticated token with permission “edit launch” to be executed.

Description	
Verb	PUT
Path	/V1.0/launches/*<launchId>*
Description	Update an existing launch by Identifier
Path Parameters	(*) launchId is a launch unique identifier (STRING)
Body Parameters	Launch
Query Parameters	
Response Status Code	200 – OK Launch [:launchId] has been updated
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access forbidden 404: Error cannot update the launch 500: Database error

17.13.2.5 Delete Launches

This API requires an authenticated token with permission “delete launch” to be executed.

Description	
Verb	DELETE
Path	/V1.0/launches/*<launchId>/revision/:launchRev*
Description	Delete an existing launch by Identifier
Path Parameters	(*) launchId is a launch unique identifier (STRING) (*) launchRev is the revision of the launch
Query Parameters	
Response Status Code	200 – OK Launch [:launchId] [:launchRev] has been deleted
Response Object	
Errors	400: Bad format or invalid identifier 403 : Access fordidden 404: Error cannot delete the launch 500: Database error

17.14 Launch Keywords

17.14.1 Overview

17.14.2 REST API

17.14.2.1 Get All Launch Keywords

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/launch-keywords
Description	Get a list of all available launch keywords
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Launch Keywords
Errors	

17.14.2.2 Get Launch Keywords

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/ launch-keywords/*<launchKeywordId>*
Description	Get a launch keyword by Identifier
Path Parameters	(*)launchKeywordId is a launch keyword unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Launch Keyword
Errors	404: Launch Keyword not found

17.15 Menu Bars

17.15.1 Overview

17.15.2 REST API

17.15.2.1 Get All Menu Bars

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/menu-bars
Description	Get a list of all available menu bars
Path Parameters	
Query Parameters	

Response Status Code	200 – OK
Response Object	List of Menu Bars
Errors	

17.15.2.2 Get Menu Bars

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/ menu-bars/*<menuBarId>*
Description	Get a Menu Bar by Identifier
Path Parameters	(*)menuBarId is a menu bar unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Menu Bar
Errors	404: Menu Bar not found

17.16 Menu Items

17.16.1 Overview

17.16.2 REST API

17.16.2.1 Get All Menu Items

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/menu-items
Description	Get a list of all available menu items
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Menu Items
Errors	

17.16.2.2 Get Menu Items

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/ menu-bars/*<menuItemId>*
Description	Get a Menu Item by Identifier
Path Parameters	(*)menuItemId is a menu Item unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Menu Item
Errors	404: Menu Item not found

17.17 Modules

17.17.1 Overview

17.17.2 REST API

17.17.2.1 Get All Modules

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/modules
Description	Get a list of all available modules
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Modules
Errors	

17.17.2.2 Get Module

This API requires an authenticated token to be executed.

Description	
Verb	GET
Path	/V1.0/ modules/*<moduleId>*
Description	Get a Module by Identifier
Path Parameters	(*)moduleId is a module unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Module
Errors	404: Module not found

17.18 Reports

17.18.1 Overview

17.18.2 REST API

17.18.2.1 Export Report

This API requires an authenticated token with permission “export report” to be executed.

Returns a report in PDF format of the content at an URI of the Unified Console. It require a query parameter ‘uri’ to indicate the page to rasterize (it could be a workspace or a view). If it is a workspace in the uri, all its views will be rasterized in pages. It also support optional settings to customize the orientation: portrait or landscape(default), format: A4, A3, B1… default is A4, margins to integrated on top, bottom, left and right (default is none)

Description	
Verb	GET
Path	/V1.0/report
Description	Get a PDF report
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	Report
Errors	

17.19 Export

17.19.1 Overview

17.19.2 REST API

17.19.2.1 Export Data

Export all data from a query (URI) in CSV.

Description	
Verb	GET
Path	/V1.0/export-data
Description	Get a CSV file
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	Report
Errors	

17.20 Themes

17.20.1 Overview

17.20.2 REST API

17.20.2.1 Get All Themes

Description	
Verb	GET
Path	/V1.0/themes
Description	Get a list of all available themes
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	List of Themes
Errors	

17.20.2.2 Get Theme

Description	
Verb	GET
Path	/V1.0/ themes/*<themeId>*
Description	Get a Theme by Identifier
Path Parameters	(*)themeId is a theme unique identifier (STRING)
Query Parameters	
Response Status Code	200 – OK
Response Object	Theme
Errors	404: Theme not found

17.21 User Preferences

17.21.1 Overview

17.21.2 REST API

17.21.2.1 Get User Preferences

Description	
Verb	GET
Path	/V1.0/ user-preferences
Description	Get global user preferences defined for the platform
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	User Preferences
Errors	404: User Preferences not found

17.22 Publish Subscribe

17.22.1 Overview

17.22.2 REST API

17.22.2.1 Get Notifications

Description	
Verb	GET
Path	/V1.0/notifications
Description	Get a set of notifications the user can subscribe to
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	Set of notifications

17.23 Platform Statistics

17.23.1 Overview

17.23.2 REST API

17.23.2.1 Get Statistics for Views

Description	
Verb	GET
Path	/V1.0/statistics/views
Description	Get statistics on Views used since the last collection
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	Statistics

17.23.2.2 Get Statistics for Workspaces

Description	
Verb	GET
Path	/V1.0/statistics/workspaces
Description	Get statistics on Workspaces used since the last collection
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	Statistics

17.23.2.3 Get Statistics for Plugins

Description	
Verb	GET
Path	/V1.0/statistics/plugins
Description	Get statistics on Plugins used since the last collection
Path Parameters	
Query Parameters	
Response Status Code	200 – OK
Response Object	Statistics

Chapter 18 Client Services

UOC Server has several common services available on the client part for integrators or developers of widgets. These common services are Angular JS modules that can be injected into your widgets to ease implementation. They are not mandatory to use but ease the integration.

Several common client services are available to build additional components:

- Data Exchange Services
- Data Access Services
- Action Services
- User Preferences Services
- Language Configuration Services
- Launches Services
- Map Configuration Services
- Theme Configuration Services
- Message Notifier Services
- Package Access Services
- Permission Services
- Audio Services

18.1 Common Events

Several JavaScript events are send / received by UOC and its components.

18.1.1 $translateChangeEnd

This event is emit by the $translate module and notify all components that the new translation has been loaded and ready to be used. So, it is possible to reload some strings and redraw component if needed.

```
$rootScope.$on('$translateChangeEnd', function() {
    $translate($scope.translationIds).then(function(translations) {
        $scope.translations = translations;
        …
        // reload all your string stored in your JavaScript
        // refresh and redraw here the component (if needed)
    });
});
```

18.1.2 webgui.dataExchange

This event is received when a parameter (listen as input parameter) has been updated, and it may require processing by a component to stay up to date.

```
// manage changes in input parameters
$scope.$on('webgui.userLoggedIn', function() {
      …
```

```
      // the user just log in, implement a special processing here if
needed
});
```

18.1.3 webgui.widgetRefresh

This event is received by a widget when it requires a refresh. It could be an automatic refresh timer or a manual refresh operation from the widget toolbar.

```
// Received when the widget must be refreshed (e.g. when clicking on the
refresh button)
$scope.$on(events['webgui.widgetRefresh'], function() {
      …
      // the widget can implement the refresh processing here
});
```

18.1.4 webgui.widgetCancelRefresh

This event is received by a widget when it requires to cancel an pending refresh (i.e. a pending quiery). It could be an automatic operation or a manual refresh operation from the widget toolbar. Usually a refresh button from the widget toolbar becomes a Cancel button until the end of the processing.

```
// Received when the refresh processing need to be cancelled (e.g. when
clicking on the cancel button)
$scope.$on(events['webgui.widgetCancelRefresh'], function() {
      …
      // the widget can implement the refresh processing here
});
```

18.1.5 webgui.userLoggedIn

This event is received by a component when the user logged in. You can listen this changes in case you need to apply a special processing for login. The authentication module is in charge of broadcasting this event.

```
// manage the event userLoggedIn
$rootScope.$on('webgui.userLoggedIn', function() {
      …
      // the user just log in, implement a special processing here if
needed
});
```

18.1.6 webgui.userLoggedOut

This event is received by a component when the user logged out. You can listen this changes in case you need to apply a special processing for logout. The authentication module is in charge of broadcasting this event and also redirect to the login page.

```
// manage the event userLoggedOut
$rootScope.$on('webgui.userLoggedOut', function() {
      …
      // the user just log out, implement a special processing here if
needed
});
```

18.1.7 webgui. widgetResize

This event is received when the widget need to resize, e.g. when the size of the view changed. For example, when the data selection tool is open.

```
// Triggered when the size of the view changes, for example when opening
the data selection menu. It is not triggered on window resize, only on
some specific cases
$scope.$on(events['webgui.widgetResize'], function() {
 });
```

18.1.8 webgui.widget.configurationChanged

This event is received when the configuration of the widget has been changed an may need additional processing to redraw the widget. A typical usage is to refresh graphically the widget to make sure changes done in configuration will be visible dynamically.

```
//Triggered when the configuration of a widget has been changed using
the configuration modal
$scope.$on(events['webgui.widgetConfigurationChanged'],
function(/*event, oldConfiguration, newConfiguration*/) {
       redraw();
});
```

18.1.9 webgui.themeChanged

This event is received when the theme has been dynamically changed, and it may require additional processing by a component. A typical usage is to dynamically apply CSS changes to the component using the client service: themeConfigServices. See 18.9 Theme Configuration Services

```
// Triggered when the UOC theme has been changed
$scope.$on(events['webgui.themeChanged'], function() {
      // dynamically apply the changes in the CSS
```

```
     themeConfigService.applyThemeCSS('xxx-my-widget');
});
```

18.1.10 webgui.notificationCreated

This event is received when a new notification has been displayed in the message notification. A component can add its own processing if needed.

```
// Triggered when a new notification is received

$scope.$on(events['webgui.notificationCreated'], function(event,
notification) {
     // notification contains the notification definition added
     doVisualProcesssing(notification.level)
});
```

18.1.11 webgui.notificationUpdated

This event is received when an existing notification update has been received and replace a previous one (e.g. progress steps in percentage...). A component can add its own processing if needed.

```
// Triggered when a notification update is received

$scope.$on(events['webgui.notificationUpdated'], function(event,
notification) {
     // notification contains the notification definition updated
     doVisualProcesssing(notification)
});
```

18.1.12 webgui.notificationDeleted

This event is received when an existing notification has been received and updated a previous notification (e.g. progress steps in percentage...). A component can add its own processing if needed.

```
// Triggered when an existing notification is deleted

$scope.$on(events['webgui.notificationUpdated'], function(event,
notification) {
     // notification contains the notification definition deleted
     $scope.removeNotification(notification);
```

```
});
```

18.1.13 webgui.messageNotifierService

This event is received when a new notification has been submitted and the component used the messageNotifierService.subscribe to listen these notifications changes. A component can add its own processing if needed. It is useful to process data notification that is not displayed in the message notifications window.

```
// Triggered when a new notification is received (useful handle data
notification in component)


$scope.$on(events['webgui.messageNotifierService], function(event,
notification) {
     // notification contains the notification definition
     doDataProcessing(notification);
});
```

18.1.14 webgui.actionRun

This event can be broadcasted to $rootScope send to execute an action. The Action Services listen this event to perform an action. In some cases, it is not possible to use directly a call to the API of the service and it is an option to use a JavaScript event. See 6.37 Action for details on the action definition.

```
// Broadcast to execute a given action


this.setAction = function(actionDefinition) {
       $rootScope.$emit(events['webgui.actionRun'], actionDefinition);)
};
```

18.2 Data Exchange Services

18.2.1 Overview

Data Exchange Service is an inter-communication service that allow components (widgets, modules…) to exchange data between components. This singleton service manages a shared memory in the web browser (client side). Every component can publish a key / value named “parameter”. An input parameter is a parameter read from this service and an output parameter is a parameter wrote in this service.

A parameter can be a default value, graphical selections, data selections, filters, etc. It is a basic data type or a complex objet (json, objet, array …)

Each parameter is associated to an expiration scope and UOC engine will manage cleanup based on this expiration scope (global or workspace). Default scope is usually workspace.

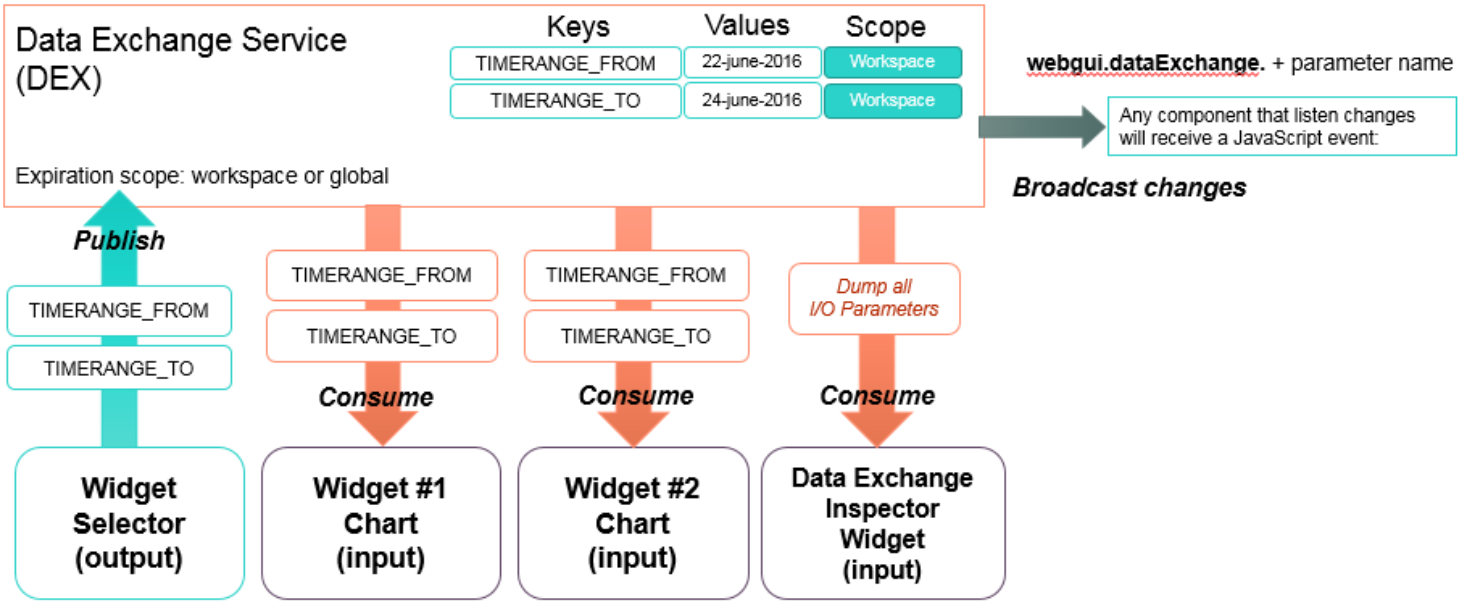


Figure 94: Data Exchange Services Overview

This service is also in charge of notification when a parameter change and allow components to refresh their queries. This service will broadcast a JavaScript event change (**webgui.dataExchange**) with the information of the parameter key that has been updated. See details in the Common Events in 0

webgui.dataExchange

Example:

- A widget W1 uses the parameter ‘CUST123’ (input parameter) and listen all real-time changes of this parameter.
- A widget W2 publish a new value to the parameter ‘CUST123’ (output parameter).
- The Data Exchange Services in charge of i/o parameters stores the new value and send a JavaScript event “webgui.dataExchange.CUST123”
- Widget W1 receives the event “webgui.dataExchange.CUST123” and queries again the data to the plugin using this new information.

NOTE: Each widget descriptor defines the official parameters in input and output supported by the widget. See Error! Reference source not found. Error! Reference source not found.

18.2.2 Expiration scopes

Two expiration scope are available: Global or Workspace

Expiration Scope	Description	Example of usage
SCOPE.GLOBAL	Global scope is seen as an application scope and it is linked to the end user session. These parameters will only be cleanup when the user logout of the unified console	Global parameter can be stored in the Data exchange service like a customer identifier selected at the beginning of the application and shared cross workspaces until the user logout
SCOPE.WORKSPACE (default scope)	Workspace scope is linked to a workspace. These parameters will be cleanup when the user close the workspace or change to another workspace.	All parameters only used by a dedicated workspace. It could be the current search to remind, the last selection of a given table … that only impact the open workspace.

18.2.3 HPE Data Exchange Inspector Widget

A special widget Data Exchange Inspector is available for troubleshooting the Data Exchange Services. This widget is very useful for View Designers, integrators and widget developers. It displays in real-time the contents of the global memory in charge of exchanging data between components. Every parameters set in the Data Exchange Service area is listed in an table and displayed by this widget.

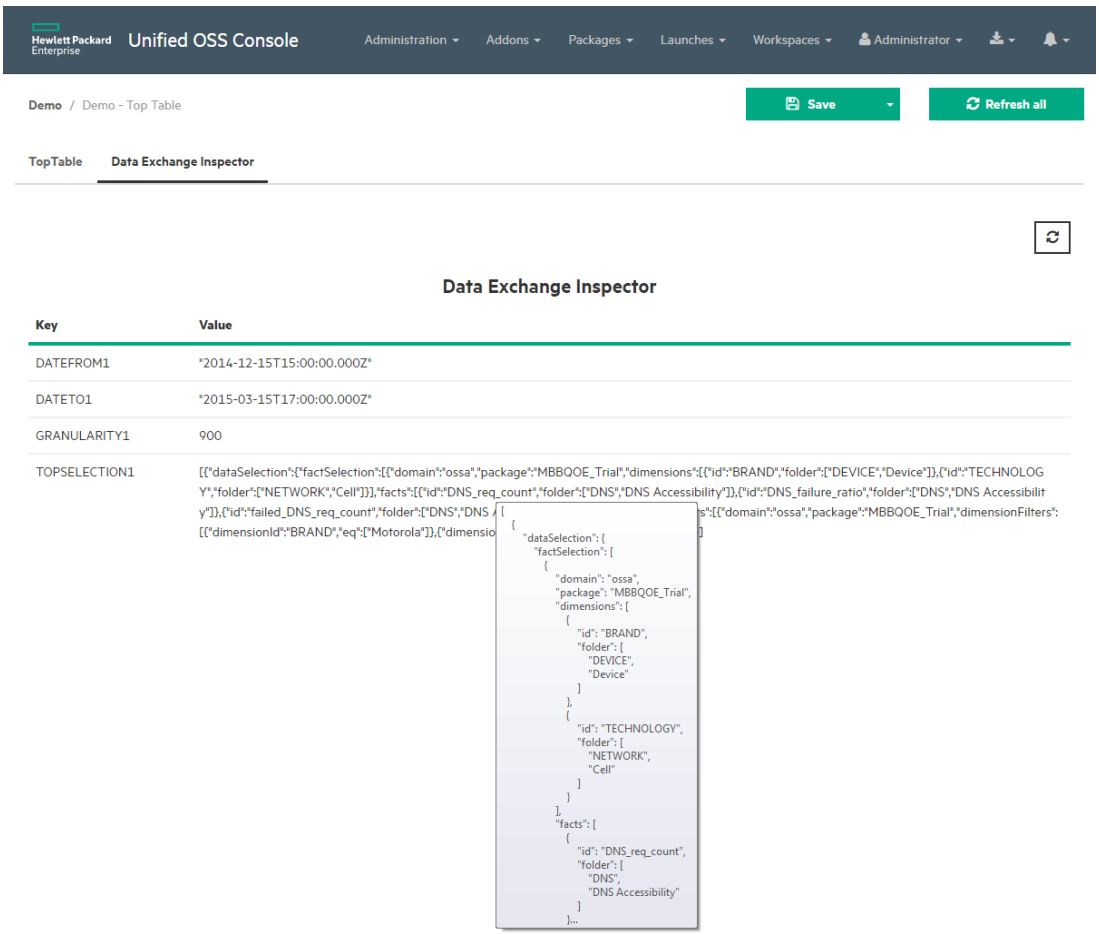


Figure 95 – HPE Data Exchange Inspector Widget (SDK)

18.2.4 Methods

18.2.4.1 Get a parameter

```
getData(name)
```

Returns the object parameter identified with its name stored in the data exchange service.

Parameter	Type	Details
Name	String	Name of the parameter

NOTE: Parameter name is not case sensitive

18.2.4.2 Set a parameter

```
setData(name, value, sendEvent, scope)
```

Stores a named parameter and its value in the data exchange service.

Parameter	Type	default	Details
Name	String		Name of the parameter
Value	Object		Value of the parameter. It can be a simple data type (string, number, boolean…) or a json object.
sendEvent *(optional)*	boolean	true	If true or undefined, the data exchange service will broadcast an event to notify about this change.
Scope *(optional)*		SCOPE.WORKSPACE	Expiration scope to used :Global or Workspace See 18.2.2 Expiration scope

NOTE: Parameter name is not case sensitive

18.2.4.3 Set a list of parameters

```
setDataList(outputs, scope)
```

Stores a named parameter and its value in the data exchange service.

Parameter	Type	default	Details
outputs	object		List of input parameters See 6.22 Input Parameters
Scope *(optional)*		SCOPE.WORKSPACE	Expiration scope to used :Global or Workspace See 18.2.2 Expiration scope

NOTE: Parameter name is not case sensitive

Example of outputs definition in JSON

```
"outputs": [{
        "name": "DATEFORM07",
        "defaultValue": "2016-02-16T00:00:00Z"
    }, {
        "name": "DATETO07",
        "defaultValue": "2016-02-20T00:00:00Z"
    }, {
        "name": "GRANULARITY07",
        "defaultValue": 900
    }]
```

18.2.4.4 Get all parameters

```
getAll()
```

Returns the complete list of parameters stored in the data exchange service.

18.2.4.5 Delete a parameter

```
deleteData(name)
```

Delete the object parameter identified with its name stored in the data exchange service.

Parameter	Type	Details
Name	String	Name of the parameter

NOTE: Parameter name is not case sensitive

18.2.5 Example

```
define([
…
      // Dependencies definition. We need to load the DEX service using
requireJS
      'components/data-exchange/data-exchange-services',
], function() {
      'use strict';
…
      // angular will inject this service with the dependency injection
      var myController = angular.module('myControllers',
['dataExchangeServices']);
      myControllers.controller('myController', ['dataExchangeService',
```

```
function(dataExchangeService) {
..
      // publish a output parameter CUSTOM_ID with the string value
CUST123
       dataExchangeService. setData('CUSTOM_ID', 'CUST123');


      // publish a list of parameter defined as an output parameters
       dataExchangeService. setDataList(output);


      // return the parameter named 'CUST123'
      var value = dataExchangeService.getData('CUSTOM_ID');     // return
'CUST123'


      // return all parameters in JSON in a list
       var all = dataExchangeService.getAll();


      // delete from the Data Exchange Service a parameter named
'CUSTOM_ID'
       dataExchangeService.deleteData('CUSTOM_ID');
};
```

```
define([
        'angular',
        'components/data-exchange/data-exchange-services',
], function(angular) {
        'use strict';
…
        var trainingWeatherControllers = angular.module('trainingWeatherControllers', ['dataExchangeServices']);

  trainingWeatherControllers.controller('trainingWeatherController', ['$scope', 'dataExchangeService',
    function($scope, dataExchangeService) {
..
        dataExchangeService. setData ('key', 'value');
        dataExchangeService. setDataList (output);
        var value = dataExchangeService.getData('key');

        var all = dataExchangeService. getAll();
        dataExchangeService. deleteData ('key');
};
```

view.json

```
"outputs": [{
      "name": "DATEFORM07",
      "defaultValue": "2015-02-16T00:00:00Z"
   }, {
      "name": "DATETO07",
      "defaultValue": "2015-02-20T00:00:00Z"
   }, {
      "name": "GRANULARITY07",
      "defaultValue": 900
   }
```

Figure 96: Example of Data Exchange Services

18.3 Data Access Services

18.3.1 Overview

Data Access Service is a major key service in charge of collecting the data from data server through the communication provided by the plugin. This intelligent service is able to manage a list of requests and uses the definition of the widget and input parameters:

- The full definition of the widget
- Dimension or object data selection
- Dimension or object filter definition
- Top filter definition
- Input parameters (WIDGET_SELECTIONS) published by other components. See 6.25

This service is very powerful and useful. It extracts from the widget, its configuration (data selection, data filter, top filter, sorting…) and is able to manage all information that impacts the queries (e.g. start time, end time, granularity, WIDGET_SELECTIONS parameters from other widgets, and build the correct URLs to contact the RESTAPI of the plug in managing the value pack (get data, execute operations, etc.). It takes care of the http request (one or multiple) in asynchronous mode and return a list of pending request (deferred). These deferred objects then can be handled by the widget to display the expected data.

IMPORTANT: Data Access Services used by all HPE Core widget to query data server based on the value pack definition.

18.3.2 Methods

18.3.2.1 Get data from the data server

```
getData(widget, delay, useCache, selections)
```

Query the data server through the plugin and return an array of deferred. This method validates data selection, data filters, build the URL to target the right plugin in charge of the value pack defined in the data selection.

Parameter	Type	default	Details
widget	Object		Widget definition in JSON containing the data selection, data filter, top filter, I/o parameters…
Delay *(optional)*	integer	0	Optional timeout in milliseconds before the promise is aborted
useCache *(optional)*	boolean	true	Boolean to indicate if the HTTP cache is to be used or not for this request.
Selections *(optional)*	Object		Specify additional data selections / data filters to be apply on top of the current data selections for the query of the widget.

It returns an array of deferred. Promises should be handled with $q.all method.

NOTE: A promise represents the result of an asynchronous operation (e.g. a value that is not yet known). A promise is in one of three different states:

- pending - *The initial state of a promise.*
- fulfilled - *The state of a promise representing a successful operation.*
- rejected - *The state of a promise representing a failed operation.*

A deferred represents a chainable utility object with methods to register multiple callbacks into callback queues, invoke callback queues, and relay the success or failure state of any synchronous or asynchronous function.

Read more details about promise and deferred at http://documentup.com/kriskowal/q/

18.3.2.2 Get object type definition

```
getTypeDefinitions(widget, delay, useCache, selections)
```

Returns object type definition defined in the data selection. This definition describes the content of the object type (properties, data types ...).

Parameter	Type	default	Details
widget	Object		Widget definition in JSON containing the data selection, data filter, top filter, I/o parameters...
Delay *(optional)*	integer	0	Optional timeout in milliseconds before the promise is aborted. 0 means infinite.
useCache *(optional)*	boolean	true	Boolean to indicate if the HTTP cache is to be used or not for this request.
Selections *(optional)*	Object		Specify additional data selections / data filters to be apply on top of the current data selections for the query of the widget. It is only used for very specific need.

18.3.2.3 Get form definition

```
getForm(widget, delay, useCache, selections)
```

Returns form definition for a given object type defined in the data selection. This form will be used to generate the GUI form used by widget form, widget search, etc.

Parameter	Type	default	Details
widget	Object		Widget definition in JSON containing the data selection, data filter, top filter, I/o parameters...
Delay *(optional)*	integer	0	Optional timeout in milliseconds before the promise is aborted. 0 means infinite.
useCache *(optional)*	boolean	true	Boolean to indicate if the HTTP cache is to be used or not for this request.
Selections *(optional)*	Object		Specify additional data selections / data filters to be apply on top of the current data selections for the query of the widget. It is only used for very specific need.

18.3.2.4 Get objects from the data server

```
getObjects(widget, delay, useCache, selections)
```

Returns objects from the data server that match the data selection and the optional filtering setting.

Parameter	Type	default	Details
widget	Object		Widget definition in JSON containing the data selection, data filter, top filter, I/o parameters…
Delay *(optional)*	integer	0	Optional timeout in milliseconds before the promise is aborted. 0 means infinite.
useCache *(optional)*	boolean	true	Boolean to indicate if the HTTP cache is to be used or not for this request.
Selections *(optional)*	Object		Specify additional data selections / data filters to be apply on top of the current data selections for the query of the widget. It is only used for very specific need.

18.3.2.5 Get Uniform Resource Identifiers (URIs)

```
getDataUris(widget, selections)
```

Returns a list of URIs used to execute requests.

Parameter	Type	default	Details
widget	Object		Widget definition in JSON containing the data selection, data filter, top filter, I/o parameters…
Selections *(optional)*	Object		Optional WIDGET_SELECTION parameters to specify additional data selections / data filters to be merged with the one from the widget. It is only used for very specific need.

NOTE: Read more details about the generic syntax of the URIs at http://www.ietf.org/rfc/rfc2396.txt

Example of HTTP request built to collect data from OSS analytics plugin

http://localhost:3000/V1.0/domains/ossa/packages/MBBQOE/facts/volume_up_sum/dims/TECHNOLOGY?begin=1410912000000&end=1411084800000&granularity=900&batchsize=100000

18.3.3 Example

The following example getData or getObjects from the data server using the data selection definition of the widget. It leverage $q to manage promises and wait for all requests to be done to process information.

NOTE: 'xxx' is the unique add-on identifier and 'myWidget' is the unique widget identifier used with our code generation tools

```
define([
    'angular',
    'lodash',
    'components/data-access/data-access-services'
], function(angular, _) {
    'use strict';
    var xxxMyWidgetServices = angular.module('xxxMyWidgetServices',
['dataAccessServices']);


    xxxMyWidgetServices.service('xxxMyWidgetService', ['$q', '$log',
'dataAccessService',
        function($q, $log, dataAccessService) {


            var logger = $log.getInstance('xxxMyWidgetService');


              /**
               * Get the formatted data from server (Asynchronous method)
               *
               * @method     getData
               * @param      {object} widget the widget
               * @return     {object} data (unified format)
               */
              this.getData = function(widget) {
                // Object Type support - use getObject if the widget
manages Object Types (with typeSelection)
                // var deferreds = dataAccessService.getObjects(widget);


                //  Dims/Facts support - use getData if the widget
manages dimensions / facts (with dataSelection)
                var deferreds = dataAccessService.getData(widget);


                var promises = _.pluck(deferreds, 'promise');
                var deferred = $q.defer();


                $q.all(promises).then(function(results) {
                    if (!results.length) {
                        deferred.resolve(null);
                    } else {
                        try {
                         var formattedData;
                            // handle the results of all the requests
                            // ...
                            deferred.resolve(formattedData);
```

```
                    } catch (e) {
                        deferred.reject(e);
                    }
                }
            }).catch(function(err) {
                deferred.reject(err);
            });
            return deferred.promise;
        };
    }
]);
return xxxMyWidgetServices;
});
```

18.4 Action Services

18.4.1 Overview

Action Services is a service in charge of executing action from anywhere in the application. It supports multiple type of actions and provide a unified way of definition and configuration.

Here follows some available predefined 'ready to use' actions:

- Navigation by View ID: To navigate to a dedicated view inside a workspace (Note: a workspace is a set of views, a view is a web page) using the View identifier. This navigation is an internal navigation inside a workspace.
- Navigation by URL: To navigate to a dedicated URL. This navigation is intended for navigation outside a workspace. It is possible to navigate cross workspaces / views or to an external application through an URL.
- Publish output parameter(s). This mechanism is widely used by UOC to share information between components (e.g. set a filter, select a value, force a refresh, etc.). This parameter (or list of parameters) can also drive the data request by leveraging the filtering mechanism. An action can populate additional filters and thus, require an update of the data request (e.g. click on a button that will filter automatically the request using 3G as technology)
- Object Type / Operations. This action gets the form associated to an operation and object and displays it as a popup dialog box. The end user can then complete the fields and submit the operation to the server.
- Launch. This action executes an UOC launch component (i.e. an external URL with dynamic and contextual parameters.
- Internal operations. To execute internal UOC operations like reload packages, etc.
- Notification: To send a real-time notification via the notification server to inform a user or a group of users, roles and people monitoring the same workspace, etc.
- Add-on module (i.e. custom action). This is a call to a generic component. This component is usually custom and allows to implement very specific needs. This module is a JavaScript angular component with a predefined interface. The function performing the action gets as input the information of the action.

More details on the syntax of each of them in 6.37 Action

The Action Service manages the action definition and relies on other common UOC services to execute the right processing

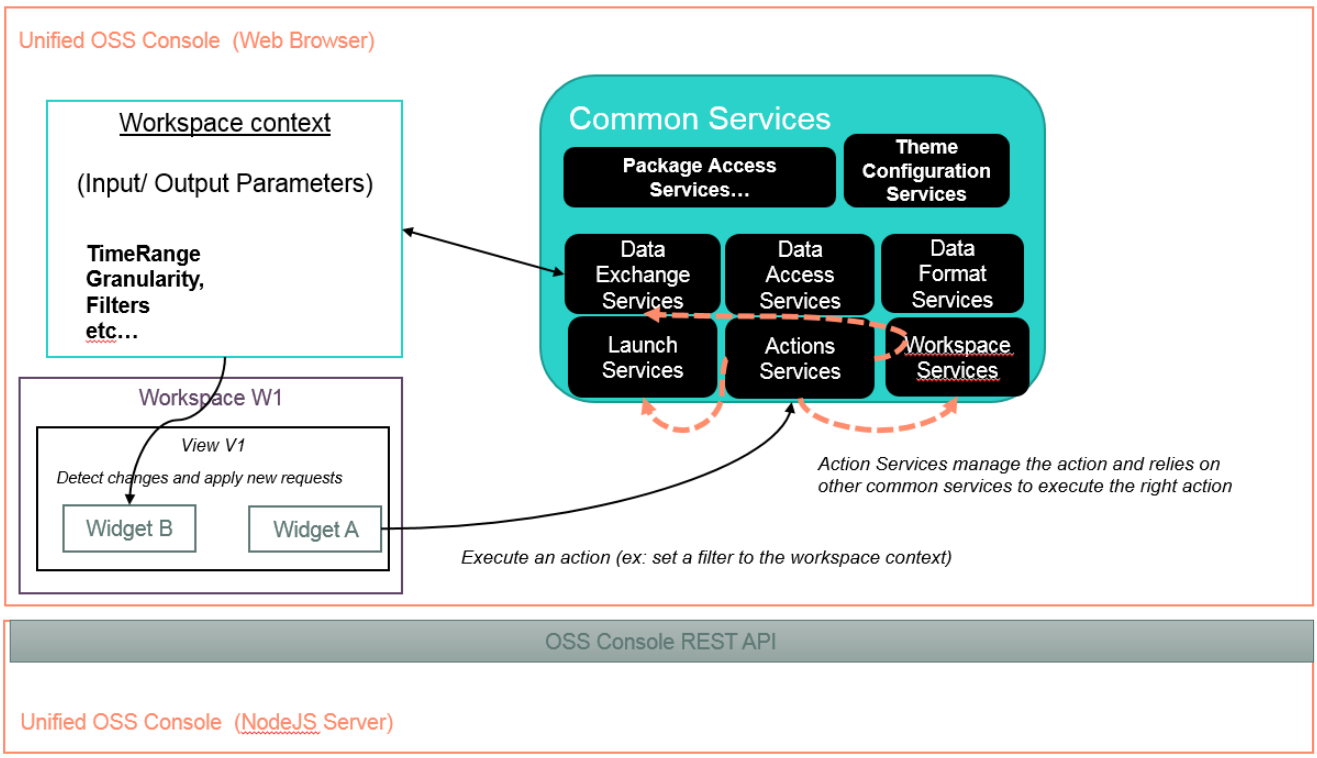


Figure 97: Action Services delegate action to other public APIs

18.4.2 Methods

`runAction(actionDefinition, options)`

Execute a given action based on its definition with optional context. S6.37 Action

Parameter	Type	default	Details
actionDefinition	Object		Definition of the action to execute
options	Object		Optional context to refine the execution. It is useful when you executing a custom add-on module and want to pass a specific context.

18.4.3 Example

```
define([
…
        'components/action/action-services'
…
    ],
    function() {
        'use strict';
        var xxxDirectives = angular.module(xxxDirectives', [
'actionServices']);     // module injection


       xxxDirectives.directive('xxx', [ 'actionService',  function(
actionService) {// service injection
                return {
```

```
…
                //Perform actions if any on click events for this widget
                if ($scope.widget.configuration &&
$scope.widget.configuration.events) {
                   var events = $scope.widget.configuration.events;
                   if (events.click) {
events.click.forEach(function(actionDefinition) {


// if several actions are configured, the component need to process them
individually.

actionService.runAction(actionDefinition);
// call the public interface to execute the action.
                    });
                }
        }
…
}
}
```

18.5 User Preferences Services

18.5.1 Overview

User preferences services is in charge of managing the user / platform preference of the connected user. This service retrieve all platform preferences customized by the user preferences.

The service manages the platform definition set by an administrator, override changes set by conditions for specific tenant identifier and/or roles identifier, and optionally information found in the SAML token set by the Identity provider, and returns the actual setting used for the connected user.

All these following settings can be accessed through the APIs.:

- Title
- Version
- Associated HTML link (when the user clicks on the title)
- Theme
- Language
- Private workspace policy (public only, public and private)
- Workspace customization (access to workspace manager, initial workspace to open, last access workspace)
- Menu Bar customization

18.5.2 Methods

18.5.2.1 Get User Identifier

```
getUser()
```

Returns user identifier of the connected user.

18.5.2.2 Get Title

```
getTitle()
```

Returns title to use for the connected user. It is useful when the Unified Console has been rebranded with a custom title.

18.5.2.3 Get Version

```
getVersion()
```

Returns the string version associated to the Unified Console. It is useful when the Unified Console has been rebranded with a custom version. <UOC:VERSION> keyword will be replaced by the current version of the product.

18.5.2.4 Get Link

```
getLink()
```

Returns the HTML link associated to the title in the main menu bar for the connected user.

18.5.2.5 Get Theme

```
getTheme()
```

Returns the theme associated to the connected user.

18.5.2.6 Get Notifications

```
getNotifications()
```

Returns the list of notifications associated to the connected user. See 6.11

18.5.2.7 Get Show / Hide for Menu Bar

```
getShowMenuBar()
```

Returns the policy set by the administrator to show or hide the main menu bar.

18.5.2.8 Get Menu Bar

```
getMenuBar()
```

Returns the menu bar identifier to use for the main menu of the Unified Console for the connected user.

18.5.2.9 Get Show / Hide for Workspace Manager

```
getShowWorkspaceManager()
```

Returns the policy set by the administrator to show / hide the workspace manager screen. It is usually coupled with an initial workspace identifier definition. See 18.5.2.10 Get Initial Workspace Manager

18.5.2.10 Get Initial Workspace Manager

```
getInitialWorkspaceManager()
```

Returns the initial workspace identifier of the workspace to open by default after login.

18.5.2.11 Get Last Access Workspace

```
getInitialWorkspaceManager()
```

Returns policy set by the administrator to support last access workspace, i.e. open directly the last opened workspace of the connected user..

18.5.2.12 Get Enable Private Workspace

```
getEnablePrivateWorkspace()
```

Returns the policy set by the administrator to support private workspace for the connected user.

18.5.3 Example

```
define([
        'angular',
        'components/user-preferences/user-preferences-services'
    ],
    function(angular) {
        'use strict';
        return angular.module('xxxControllers',
[]).controller('xxxController', ['$scope', 'userPreferencesService',

            function($scope, userPreferencesService) {

                $scope.title = userPreferencesService.getTitle();
```

```
                $scope.link = userPreferencesService.getLink();

                $scope.menuBarId = userPreferencesService.getMenuBar();
                $scope.showMenuBar
userPreferencesService.getShowMenuBar();

                $scope.isMenuBarVisible = function() {
                    return $scope.showMenuBar;
                };

            }
        ]);
    });
```

18.6 Language Configuration Services

18.6.1 Overview

Language Configuration Services is in charge of managing the supported languages of the Unified Console. This service let you know the list of supported language installed in the Unified Console platform, and let you drive dynamically the change if needed.

NOTE: The installed language list if defined by a platform administrator in the configuration file in <installdir>/server/public/conf/config.json

Example of config.json

Example of config.json
It defines US English and French and Spanish locale with “staticL10nFile” option enabled for Spanish..

```
{...
   "languages" : [{
      "name": "English",
      "languageCode": "en-us"
   }, {
      "name": "Français",
      "languageCode": "fr-fr"
   }, {
      "name": "Español",
      "languageCode": "es-es",
      "staticL10nFile": true
   }],
...}
```

This service will use the country code and language code defined by ISO 639.

NOTE: ISO 639 is a standardized nomenclature used to classify languages. Each language is assigned a two-letter (639-1) and three-letter (639-2 and 639-3), lowercase abbreviation, amended in later versions of the nomenclature. See http://www.iso.org/iso/home/standards/language_codes.htm

18.6.2 Methods

18.6.2.1 Get languages

`getLanguages()`

Returns the list of available language for the platform (name and language code).

18.6.2.2 Get Language

`getLanguageById(languageId)`

Returns the language object identified by the language code on the platform (name and language code).

Parameter	Type	Details
languageId	String	Identifier for the language (ISO-639)

18.6.2.3 Change language

`changeLanguage(languageId)`

Set and switch dynamically to the given language identified by its language code.

Parameter	Type	Details
languageId	String	Identifier for the language (ISO-639)

NOTE: The service will broadcast to all client components, a JavaScript event named "$translateChangeEnd". See Angular translate ofr API details at https://angular-translate.github.io/docs/#/guide/18_events

It also saves in the cookie the active language identifier in "userLanguage"

18.6.3 Example

```
define([
    'angular',
    'components/language-config/language-config-services'
], function(angular) {
    'use strict';


    var languageSelectorControllers = angular.module('xxxControllers',
['languageConfigServices']);


    languageSelectorControllers.controller('xxxController', ['$scope',
'languageConfigService',
        function( $scope, languageConfigService) {
```

```
            // Available languages List
            var refreshLanguages = function() {

     languageConfigService.getLanguages().then(function(languages) {
                    $scope.languages = [];
                    languages.data.list.forEach(function(language) {
                        $scope.languages.push(language);
                    });
                }).then(null, function(error) {
                    logger.error('Cannot get list of languages', error);
                });
            };
...
            // Switch dynamically language
            $scope.changeLanguage = function(langKey) {
                    languageConfigService.changeLanguage(langKey);
                }
            };
        }
    ]);
});
```

18.7 Launches Services

Please contact R&D Team to get latest official API

18.8 Map Configuration Services

18.8.1 Overview

Map Configuration Services is in charge of getting the GeoJSON file for hpe-ng-highmaps. It also manage a cache to optimize the multiple queries to the same map.

NOTE: GeoJSON is a format for encoding data about geographic features using JavaScript Object Notation (JSON). More details at http://geojson.org/geojson-spec.html

Geo JSON Map Files are stored in <installdir>/client/public/maps with the following naming convention: <mapId>.geo.json

18.8.2 Methods

18.8.2.1 Get Geo JSON map file

`getGeoJSON(mapId)`

Returns the Geo JSON map file identifier by its map identifier.

Parameter	Type	Details
mapId	String	Map Identifier to retrieve the map file

18.8.2.2 Clear Cache

`clearCache()`

Clear the cache associated to the Geo JSON map file.

18.8.3 Example

```
define([
        'angular',
        'components/map-config/map-config-services'
    ],
    function(angular) {
        'use strict';
        return angular.module('xxxControllers', ['mapConfigServices'])
            .controller(
                'xxxController', ['$scope', 'mapConfigService',
                    function($scope, mapConfigService) {

                        var drawMap = function() {
                        var geoJsonUrlDeffered =
mapConfigService.getGeoJSON('fr-fr');
                            geoJsonUrlDeffered.then(function(data) {
                                // data contains the geo json map
                                $scope.geoJsonData = data;
                            }, function(error) {
                                // Error handling here
                            })
                        };
                    }
                ]);
    });
```

18.9 Theme Configuration Services

18.9.1 Overview

Theme Configuration Services is in charge of managing the theme in Unified Console including a cache mechanism to optimize request to the server.

18.9.2 Methods

18.9.2.1 Get Themes

```
getThemes()
```

Returns the list of available themes installed on the Unified Console. See 6.18 Theme

18.9.2.2 Get Theme

```
getTheme(themeId)
```

Returns the theme associated the theme identifier on the Unified Console. See 6.18 Theme

Parameter	Type	Details
themeId	String	Unique theme identifier of the theme

18.9.2.3 Change Theme

```
changeTheme(themeId)
```

Set and switch dynamically to the given theme identified by its unique identifier.

Parameter	Type	Details
themeId	String	Unique theme identifier of the theme

NOTE: The service will broadcast to all client components, a JavaScript event named "webgui.themeChanged". It also saves in the cookie the active theme identifier in "theme"

18.9.2.4 Apply Theme CSS

```
applyThemeCSS(themeCSS)
```

This method is usually called by each component that want to apply dynamically changes after a theme changed. It is usually coupled to event change "webgui.themeChanged" to dynamically apply CSS changes found in the active theme.

Parameter	Type	Details
themeCSS	String	Unique component identifier (widget...). This identifier will be used to find specific customization of this component in the active theme.

NOTE: These CSS files are stored under each theme under <installdir>/client/addons/<addonId>/themes/<themeId>/css and follow a naming convention to reuse the widget identifier (e.g. hpe-ng-table) and .css

18.9.2.5 Clear Cache

```
clearCache()
```

Clear the cache associated to the Geo JSON map file.

18.9.3 Example

```
define([
    'angular',
    'commons/events/commons-events',
    'components/theme-config/theme-config-services'
], function(angular) {
    'use strict';
    var xxxMyWidgetControllers =
angular.module('xxxMyWidgetControllers', ['commonsEvents',
'themeConfigServices']);

    xxxMyWidgetControllers.controller('xxxMyWidgetController',
['$scope',  'themeConfigService',
        function($scope, themeConfigService) {

...

            /** Triggered when the UOC theme has been changed */
            $scope.$on(events['webgui.themeChanged'], function() {
                themeConfigService.applyThemeCSS('xxx-my-widget');
            });

            themeConfigService.applyThemeCSS('xxx-my-widget');
        }
    ]);

    return xxxMyWidgetControllers;
});
```

18.10 Message Notifier Services

18.10.1 Overview

18.10.2 Text message and variable replacement

Some of the deprecated API (info, success, warning, danger, info) uses two parameters for the notification.

- Text: The text message to send to the message notification window. It can {{}} to define variable to replace dynamically by the correct value.
- Params.: JSON object used to defined variables to replace. List of key / values.

Example of parameters:

Parameter	Value
Text	"My name is {{name}}, I am {{age}}yo"
params	"{name: 'john', age: '32' }"

Will display the following notification message: "My name is john, I am 32yo"

18.10.3 Methods

18.10.3.1 Get Notifications

```
getNotifications()
```

Returns the list of active notifications available in the message notification of the Unified Console. See 6.11

18.10.3.2 Clear Notifications

```
clear()
```

Clear the current list of active notifications. The Message Notification will be empty.

18.10.3.3 Send Notification Info

```
info(text, params)
```

Send a text message with the level information. See 18.10.2 Text message and variable replacement

Parameter	Type	Details
text	String	Text message to send to the message notification window
params	Object	JSON object (key / value) to define value to replace in variables.

IMPORTANT: This API is deprated and replaced by notify(notifications). See 18.10.3.8 Send Notification

18.10.3.4 Send Notification Success

```
success(text, params)
```

Send a text message with the level success. See 18.10.2 Text message and variable replacement

Parameter	Type	Details
text	String	Text message to send to the message notification window
params	Object	JSON object (key / value) to define value to replace in variables.

IMPORTANT: This API is deprated and replaced by notify(notifications). See 18.10.3.8 Send Notification

18.10.3.5 Send Notification Warning

```
warning(text, params)
```

Send a text message with the level warning. See 18.10.2 Text message and variable replacement

Parameter	Type	Details
text	String	Text message to send to the message notification window
params	Object	JSON object (key / value) to define value to replace in variables.

IMPORTANT: This API is deprated and replaced by notify(notifications). See 18.10.3.8 Send Notification

18.10.3.6 Send Notification Danger

```
danger(text, params)
```

Send a text message with the level danger. See 18.10.2 Text message and variable replacement

Parameter	Type	Details
text	String	Text message to send to the message notification window
params	Object	JSON object (key / value) to define value to replace in variables.

IMPORTANT: This API is deprated and replaced by notify(notifications). See 18.10.3.8 Send Notification

18.10.3.7 Send Notification Success

```
info(text, params)
```

Send a text message with the level success. See 18.10.2 Text message and variable replacement

Parameter	Type	Details
text	String	Text message to send to the message notification window
params	Object	JSON object (key / value) to define value to replace in variables.

IMPORTANT: This API is deprated and replaced by notify(notifications). See 18.10.3.8 Send Notification

18.10.3.8 Send Notification

`notify(notification)`

Send an object notification. See 6.11 Notification Composite Key

The object notification has the following definition in JSON. The composite key refers to a notification. **<install_dir>/json-schema/notification-composite-key.json**

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Notification Composite Key",
    "description": "An explanation about the purpose of this instance.",
    "properties": {
        "type": {
            "type": "string",
            "title": "Notification Type",
            "description": "Type of the notification (e.g. alert or
data)"
        },
        "id": {
            "type": "string",
            "title": "Notification Identifier",
            "description": "Identifier of the notification"
        },
        "origin": {
            "type": "string",
            "title": "Origin",
            "description": "Origin of the notification (server, plugin,
client or workspace)"
        },
        "domain": {
            "type": "string",
            "title": "Domain",
            "description": "Domain identifier where the notification is
coming from"
        },
```

```
        "package": {
            "type": "string",
            "title": "Package",
            "description": "Package identifier where the notification is
coming from"
        },
        "workspace": {
            "type": "string",
            "title": "Workspace",
            "description": "Workspace identifier where the notification
is coming from"
        }
    },
    "required": [ "type", "id","origin" ]
}
```

Notification:

Parameter	Type	Details
Notification	Object	Notification definition to send to the message notification

NOTE: The service will broadcast to all client components, a JavaScript events. See 18.1.10 webgui.notificationCreated, 18.1.11 webgui.notificationUpdated

18.10.3.9 Delete Notification

```
notify(notification)
```

Remove a notification from the message notification using its definition. See 6.11 Notification Composite Key

The object notification has the following definition in JSON. The composite key refers to a notification. **<install_dir>/json-schema/notification-composite-key.json**

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Notification Composite Key",
    "description": "An explanation about the purpose of this instance.",
    "properties": {
        "type": {
            "type": "string",
            "title": "Notification Type",
            "description": "Type of the notification (e.g. alert or
data)"
        },
        "id": {
```

```
            "type": "string",
            "title": "Notification Identifier",
            "description": "Identifier of the notification"
        },
        "origin": {
            "type": "string",
            "title": "Origin",
            "description": "Origin of the notification (server, plugin,
client or workspace)"
        },
        "domain": {
            "type": "string",
            "title": "Domain",
            "description": "Domain identifier where the notification is
coming from"
        },
        "package": {
            "type": "string",
            "title": "Package",
            "description": "Package identifier where the notification is
coming from"
        },
        "workspace": {
            "type": "string",
            "title": "Workspace",
            "description": "Workspace identifier where the notification
is coming from"
        }
    },
    "required": [ "type", "id","origin" ]
}
```

Notification

Parameter	Type	Details
Notification	Object	Notification definition to remove from to the message notification

NOTE: The service will broadcast to all client components, a JavaScript event. See 18.1.12 webgui.notificationDeleted

18.10.3.10 Subscribe to notifications

```
subscribe(scope, compositeKey, callback)
```

Remove a notification from the message notification using its definition. See 6.11 Notification Composite Key

The object notification has the following definition in JSON. The composite key refers to a notification. <install_dir>/json-schema/notification-composite-key.json

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "title": "Notification Composite Key",
    "description": "An explanation about the purpose of this instance.",
    "properties": {
        "type": {
            "type": "string",
            "title": "Notification Type",
            "description": "Type of the notification (e.g. alert or
data)"
        },
        "id": {
            "type": "string",
            "title": "Notification Identifier",
            "description": "Identifier of the notification"
        },
        "origin": {
            "type": "string",
            "title": "Origin",
            "description": "Origin of the notification (server, plugin,
client or workspace)"
        },
        "domain": {
            "type": "string",
            "title": "Domain",
            "description": "Domain identifier where the notification is
coming from"
        },
        "package": {
            "type": "string",
            "title": "Package",
            "description": "Package identifier where the notification is
coming from"
        },
        "workspace": {
```

```
            "type": "string",
            "title": "Workspace",
            "description": "Workspace identifier where the notification
is coming from"
        }
    },
    "required": [ "type", "id","origin" ]
}
```

Notification

Parameter	Type	Details
scope	Object	Scope object of the component (e.g. widget) who is interested in this subscription
compositeKey	Object	Unique identifier for a subscription using several properties of the notification definition. See 6.11 Notification Composite Key
Callback	Function	Function to call when the notification is received

NOTE: The service will broadcast to all client components, a JavaScript event.

18.10.4 Example

```
define([
    'angular',
    'components/message-notifier/message-notifier-services'
], function(angular) {
    'use strict';
    var xxxControllers = angular.module('xxxControllers', [
'messageNotifierServices']);


    xxxControllers.controller('xxxController', ['$scope',
'messageNotifierService',
        function($scope, messageNotifierService) {


            var compositeKey = "[{
                'origin'': 'plugin',
                'domain': 'hpe',
                'package': 'MBBQOE',
                'type': 'data',
                'id': 'real-time-downlink'
            }]";
           messageNotifierService.info(“Display an information message
in console”);
           // We are notifying an alert
           messageNotifierService.notify(notificationToSend);
            // subscribe to messages corresponding to the notification
in the widget configuration
            messageNotifierService.subscribe($scope, compositeKey,
function(notification) {
                addMessage(notification);
            });
        }
    ]);
    return xxxControllers;
});
```

18.11 Package Access Services

18.11.1 Overview

Package Access Services is in charge of the available packages and allow client components to access to package definitions (e.g. metadata of dimensions, facts, aggregation tree, etc.)

18.11.2 Methods

18.11.2.1 Get All Packages

```
getPackages()
```

Returns a promise that allow you to retrieve all available packages. This promise will return a JSON list of packages (content)

18.11.2.2 Get Package

```
getPackage(domain, packageId)
```

Returns a promise that allow you to retrieve a given package. This promise will return a package.

Parameter	Type	default	Details
domain	String		Unique domain identifier. It is usually liked to the plugin identifier.
PackageId	String		Unique Package identifier to retrieve

18.11.2.3 Get Dimension

```
getDimension(domain, packageId, folder, dimensionId)
```

Returns a promise that allow you to retrieve the definition of a given dimension of a specific package. This promise will return a dimension definition.

Parameter	Type	default	Details
Domain	String		Unique domain identifier. It is usually liked to the plugin identifier.
PackageId	String		Unique Package identifier to retrieve.
Folder	String		Folder that belong to the dimension identifier. It could be an array of folders separated by comma to identify a folder and its sub-folders. e.g. [folder1, folder2]
DimensionId	String		Dimension identifier defined in a package.

18.11.2.4 Get Fact

```
getFact(domain, packageId, factId, folder)
```

Returns a promise that allow you to asynchronous retrieve the definition of a given fact of a package. The fact identifier may not unique and require the specification of a folder to retrieve the right fact.

This promise will return a fact definition.

Parameter	Type	default	Details
Domain	String		Unique domain identifier. It is usually liked to the plugin identifier.

PackageId	String		Unique Package identifier to retrieve.
Folder	String		Folder that belong to the fact identifier. It could be an array of folders separated by comma to identify a folder and its sub-folders. e.g. [folder1, folder2]
FactId	String		Fact identifier defined in a package.

18.11.2.5 Get Unique Fact

```
getUniqueFact(domain, packageId, factId)
```

Return a promise that allow you to asynchronous retrieve the definition of a given fact of a package. The fact identifier is unique and do not require a folder to identify the right fact.

This promise will return a fact definition.

Parameter	Type	default	Details
Domain	String		Unique domain identifier. It is usually liked to the plugin identifier.
PackageId	String		Unique Package identifier to retrieve.
FactId	String		Unique Fact identifier defined in a package.

18.11.2.6 Get Fact Aggregation Tree

```
getFactAggregationTree(domain, packageId, factId)
```

Return a promise that allow you to retrieve the definition of an aggregation tree for a given fact of a specific package. This promise will return a fact definition.

Parameter	Type	default	Details
Domain	String		Unique domain identifier. It is usually liked to the plugin identifier.
PackageId	String		Unique Package identifier to retrieve.
FactId	String		Fact identifier defined in a package.

NOTE: An aggregation tree describe the aggregated computing tree used to compute some fact.

18.11.3 Example

```
define([
        'components/package-access/package-access-services'
    ],
    function() {
        'use strict';
        var xxxServices = angular.module('xxxServices',
['packageAccessServices']);
        xxxServices.service('xxxService', ['packageAccessService',
            function(packageAccessService) {
                this.buildWidget = function(widget, objects) {
                    ...
                    // process all available packages
packageAccessService.getPackages().then(function(valuePacks) {


                       // valuePacks contains a array of all available
packages
                        doProcessing(valuePacks);
                        deferred.resolve(valuePacks);
                    }).then(null, function(error) {
                        deferred.reject(error);
                    });


                    // retrieve a dimension metadata
                    packageAccessService.getDimension(domain, pck,
_folder, _id).then(function(data) {
                        data ? closure() : callback(new Error('Dimension
not found [domain=' + domain + ', package=' + pck + ', folder=' +
JSON.stringify(_folder) + ', id=' + _id + ']'));
                    });


                    // retrieve a fact metadata
                    packageAccessService.getUniqueFact(domain, pck,
_id).then(function(data) {
                        data ? closure() : callback(new Error('Fact not
found [domain=' + domain + ', package=' + pck + ', id=' + _id + ']'));
                    });
                };
            }
        ]);
    });
```

18.12 Permission Services

18.12.1 Overview

This service is in charge of validating that the connected user has the right level of permission in a programmatic way. So a custom component can integrate a RBAC validation during HTML generation or in a JavaScript processing.

A permission is defined with a Verb and an object like 'save workspace', 'edit view', 'delete launch', 'create category', etc.

NOTE: Available permissions are defined in a permission configuration file in <installdir>/data/permissions.

The file permissions.json defines UOC permissions (identifier, name, description, category…)

18.12.1.1 Permission List on an object

It is possible to define a list of permissions defining verb and object separated by comma like 'save workspace, delete workspace' that authorize only user that has save OR delete permission on workspace object.

18.12.1.2 Any permission on an object

Wildcard ('*') for verb is supported and mean 'any' verb is authorized (at least one associated to the right object must be defined in the user profile)

18.12.1.3 All permissions on an object

The ampersand char ('&') verb is also supported and mean 'all' verb must be defined. E.g. '& category' means the connected user must have in his user profile all available verb (all permissions) like 'browse, create, delete, and edit category'. A connected user that will not have all the required permissions will be rejected.

18.12.2 Methods

18.12.2.1 Has Permission

```
hasPermission(values)
```

Return a boolean to validate the connected user has the correct permissions that is required.

Parameter	Type	default	Details
Values	String Verb + object separated by comma		List of permission identifiers that is required and need to be checked.

18.12.2.2 Has Role

```
hasRoles(values)
```

Return a boolean to validate the connected user has the correct roles that is required.

Parameter	Type	default	Details
Values	Arrary of role identifier (string)		List of role identifiers that is required and need to be checked.

18.12.3 Directives

Usually component that want to manage some condition rendering based on the user profile will inetrgate the angular directive provided by UOC to generate or not some HTML blocks.

This directives works the same way as the API directly in the HTML.

18.12.3.1 <has-permission>

<has-permission> allow to protect a part of HTML and validate before rendering the HTML that the connected user has the right level of permissions. If the user does not have the right this part will not be generated in the web browser. It allows an add-on developer to provide dynamically a rendering based on RBAC.

This directive can be used as an element

```
<has-permission values='...'"> </has-permission>
```

Or it can be used as an attribute

```
<div data-has-permission="...">...</div>
```

Where '...' represents the list of required permissions.

18.12.3.2 <has-role>

<has-role> allow to protect a part of HTML and validate before rendering the HTML that the connected user has the right level of roles. If the user does not have the right this part will not be generated in the web browser. It allows an add-on developer to provide dynamically a rendering based on RBAC.

This directive can be used as an element

```
<has-roles values="..."> </has-roles>
```

Or it can be used as an attribute

```
<div data-has-roles="...">...</div>
```

Where '...' represents the list of required role identifier.

18.12.4 Examples

18.12.4.1 HTML

Example from the HPE menu Item administration

```
<nav class="navbar navbar-default" role="navigation" hpe-no-report>
    <div class="container-fluid">
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav navbar-right">
                <has-permission values="* user,* role,* category,browse
launch,create launch,edit launch,delete launch,* launch_category">
                    <hpe-menu-item-administration></hpe-menu-item-
administration>
```

```
                </has-permission>
                <has-permission values="browse layout,browse
widget,browse plugin, browse menu_bar, browse menu_item, browse theme,
browse launch_keyword">
                    <hpe-menu-item-addons></hpe-menu-item-addons>
                </has-permission>
                <has-permission values="browse workspace">
                    <hpe-menu-item-workspaces></hpe-menu-item-
workspaces>
                </has-permission>


                <has-permission values="export data">
                    <hpe-menu-item-exports></hpe-menu-item-exports>
                </has-permission>
            </ul>
        </div>
    </div>
</nav>
```

Example from the HPE widget action button to enable/disable launch

```
<div class="row">
    <div class='hpe-action-dropdown col-xs-12 col-sm-
{{hpeActionInit.sizeBtn}} col-sm-push-{{hpeActionInit.showOnSide}}'>
        <div class="btn-group btn-block margin-bottom-15">
...
            <ul class="dropdown-menu hpe-action-dropdown-menu col-xs-
12">
...
                <li data-has-role="['operator_l2','operator_l3']" ng-
if="element.type ==='launch'" ng-class="{linkSelected: actionSelected ==
element.id}" data-ng-repeat="launch in launches[element.id]" data-ng-
click="runLaunch(launch)" uib-tooltip="{{launch.description}}">
                    <a href="">
                        <i data-ng-if="launch.icon"
class="{{launch.icon}} hpe-action-items-icon"></i>  
                        <translate>{{launch.name}}</translate>
                    </a>
                </li>
            </ul>
        </div>
    </div>
</div>
```

18.12.4.2 JavaScript

```
define([
        'components/permission-handler/permission-handler-services'
    ],
    function() {
        'use strict';
        var xxxControllers = angular.module('xxxControllers',
['permissionHandlerServices']);

        xxxControllers.controller('workspaceController', ['$scope',
'permissionHandlerService',
            function($scope, permissionHandlerService) {

                ...
                $scope.hasSaveEditWks =
permissionHandlerService.hasPermission('save workspace, edit
workspace');

                $scope.hasAnyPermissionsOnPrivatePermission =
permissionHandlerService.hasPermission('* private_workspace');

                $scope.hasAllPermissionsOnViews =
permissionHandlerService.hasPermission('& views');
            }
        ]);
        return xxxControllers;
    });
```

18.13 Audio Services

18.13.1 Overview

Audio Service is a common service in charge of playing sounds and notification. It uses the audio library directories. This service relies on the HTML 5 API to play sounds and the standard audio object.

In HTML5, 3 formats are supported: MP3, OGG, WAV, but to grant a full compatibility with all web browser, it is strongly recommended to use only MP3 format. More details at http://www.w3schools.com/html/html5_audio.asp

NOTE: MPEG-1 and/or MPEG-2 Audio Layer III, more commonly referred to as MP3, is an audio coding format for digital audio which uses a form of lossy data compression.

This service is also used by the Notifications engine to play audible notification.

18.13.2 Audio Files

18.13.2.1 Location

All the usable audio file for Unified Console are stored in the audio library in MP3 format in the following path:

```
<install directory>/client/public/sounds
```

Notification audio files are stored in a dedicated directory:

```
<install directory>/client/public/sounds/notifications
```

18.13.2.2 Default Audio Files

By default, the following notification sounds are available:

- danger.mp3
- default.mp3
- disabled.mp3
- fatal.mp3
- info.mp3
- in-progress.mp3
- major.mp3
- minor.mp3
- pending.mp3
- success.mp3
- unknown.mp3
- warning.mp3

18.13.3 Methods

18.13.3.1 Play a sound

```
play(file, volume, nbRepeat, delay)
```

Play a given audio file with optional setting like repetition, volume, and delay between 2 plays.

Parameter	Type	default	Details
File	String		Filename of the audio file to play
Volume	Integer	0.5	Volume used to play the file. Value is between 0 (min) and 1 (max). 0.5 is medium.
nbRepeat *(optional)*	Integer	1	If true or undefined, the data exchange service will broadcast an event to notify about this change.
Delay *(optional)*	Integer	0	Delay between two repetitions of the audio file. Useful only with multiple repeat of the audio file.

NOTE: file is the filename of the audio file stored in the /client/public/sounds

18.13.3.2 Play an audible notification

`playAudibleNotification(notification)`

Play a given notification audio file with optional setting like repetition, volume, and delay between 2 plays.

Parameter	Type	default	Details
Notification	Object		Notification definition (see 6.11). Some sound properties can be customized (file, volume, repeat, delay). Default values will be used if not set: • If file is not defined, default audio file based on notification level will be used. • If volume is not defined, default volume is set to 0.5 (medium) • If repeat is not defined, the sound is played only 1 time and delay is disabled.

NOTE: file is the filename of the audio file stored in the /client/public/sounds/notifications.

18.13.4 Example

```
define([
        'angular',
        'components/audio/audio-services'
    ],
    function(angular) {
        'use strict';

        var messageNotifierServices =
angular.module('messageNotifierServices', ['audioServices']);
        messageNotifierServices.service('messageNotifierService',
['audioService',
            function(audioService) {
              this.notify = function(notification) {
                audioService.playAudibleNotification(notification);
```

```
            };
…         }
        ]);
        return messageNotifierServices;
    }
);
```

Chapter 19 Localization (L10N) / Internationalization (I18N)

19.1 Overview

The UOC completely supports the localization and the internationalization. The UOC Client components relies on Angular Translate to manage dynamically loading of the correct locale. See 2.7 Angular Translate (L10N)

By default, the UOC uses the locale of the web browser to set the locale to use (if available). In case the detected locale is not supported, the platform will use the default setting in the user-preferences settings (usually English).

The end user can dynamically switch the language using the language selector in the user menu of the navigation menu bar.

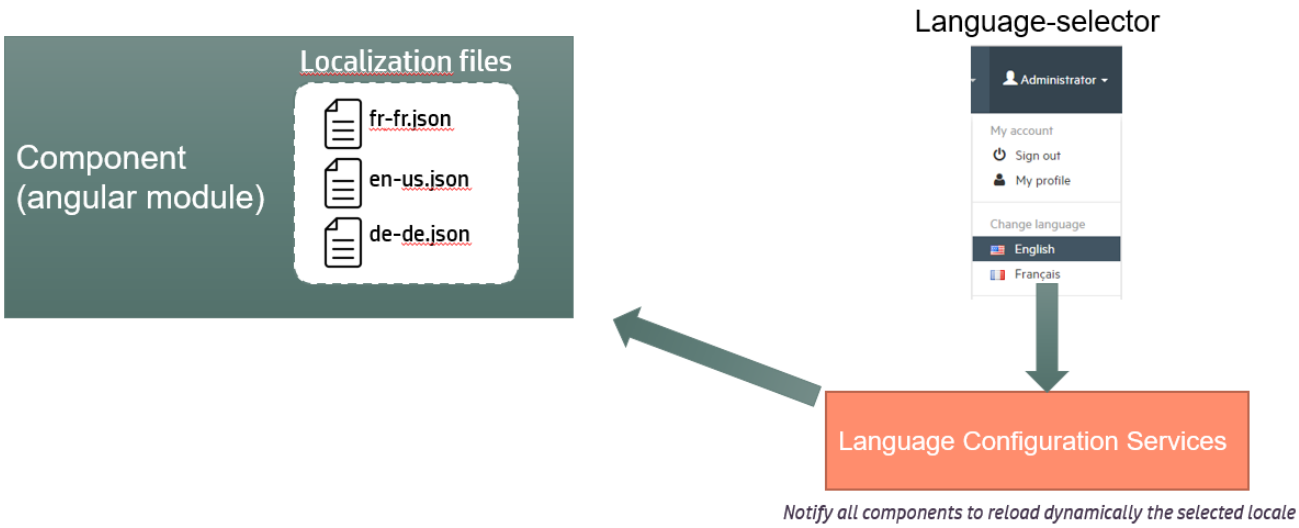


Figure 98: Localization (L10N) / Internationalization (I18N) Overview

19.2 Supported Language

UOC allows the platform administration to setup all supported language by the UOC Server and some policy for optimization.

It define the list of available language for the platform (name and language code) and an optional property to keep all localization in a single file. This is interesting to add customized languages not supported by default by UOC.

staticL10nFile is optional and its default value is false. If "staticL10nFile" value is true, L10N file will not be generated at server startup meaning that a complete L10N file must be provided and added in <install_dir>/client/l10n with the following naming rule: [languageCode].json where [languageCode] is replaced by the language code. (Example of L10N filename: es-es.json) .

Example of config.json that defines US English, French and Spanish locale (with "staticL10nFile" option enabled for Spanish)

```
{...
   "languages" : [{
      "name": "English",
      "languageCode": "en-us"
   }, {
      "name": "Français",
      "languageCode": "fr-fr"
   }, {
      "name": "Español",
      "languageCode": "es-es",
      "staticL10nFile": true
   }],
...}
```

NOTE: By default, several flags icons are available in <installDir>/client/public/images/languages

These icons of the selected language will be displayed in the menu item preference of the user, and he will be able to dynamically switch the language. If a icon is not already present in the list, it is possible to extend these icons.

IMPORTANT:. staticL10nFile property applies only if L10N optimization is enabled

All angular module integrate one or multiple L10N file with all the text to use for a dedicated locale. These file follow a strict naming convention based on the language code. One localization JSON file per supported language codes.

```
<language code>.json
```

Example:

en-us.json	fr-fr-.json
{ "XXX_MY_WIDGET": { "NO_DATA": "No data to display", "MESSAGES": { "ERRORS": { "GET_DATA_ERROR": "An error occurred while retrieving data" } } } }	{ "XXX_MY_WIDGET": { "NO_DATA": "Pas de données à afficher", "MESSAGES": { "ERRORS": { "GET_DATA_ERROR": "Une erreur s'est produite lors de la récupération des données" } } } }

19.3 Languages Optimization

The UOC platform is able to enable or disable the language loading optimization at server startup. By default this optimization is enabled but it can be disabled by setting "optimizeL10n" to false.

When L10N optimization is enabled, all language files dynamically found on the platform are concatenated in one single language file in <install_dir>/client/l10n and loaded in one request at the beginning of the application.

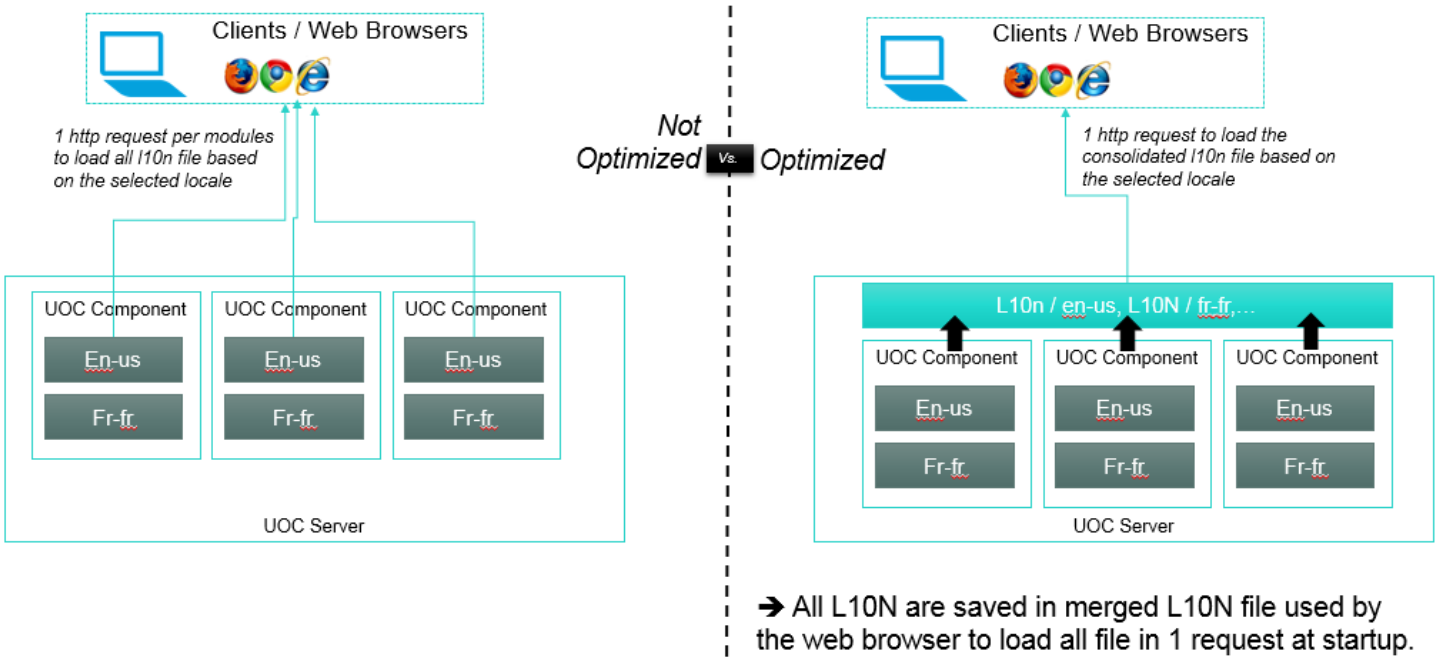


Figure 99: Language Optimization

L10N file is not generated for this language if staticL10nFile is set to true (see Installation and Configuration Guide)

19.4 Widget Localization Inspector

The Widget Localization inspector is dedicated to SDK developer to identify quickly in all his add-ons and components, all the mistake related to the L10N support. Like missing localization, invalid localization ... scanning automatically all the message file and matching the defined locale in UOC.

The widget gives a status for all add-on client components:

- Success: Localization is available and different for all locales
- Warning: Localization is available but not change between locale. It is probably a error (wrong cut/paste) and need attention.
- Missing: Localization is not available or the message is empty. It is always an error that need to be fixed.

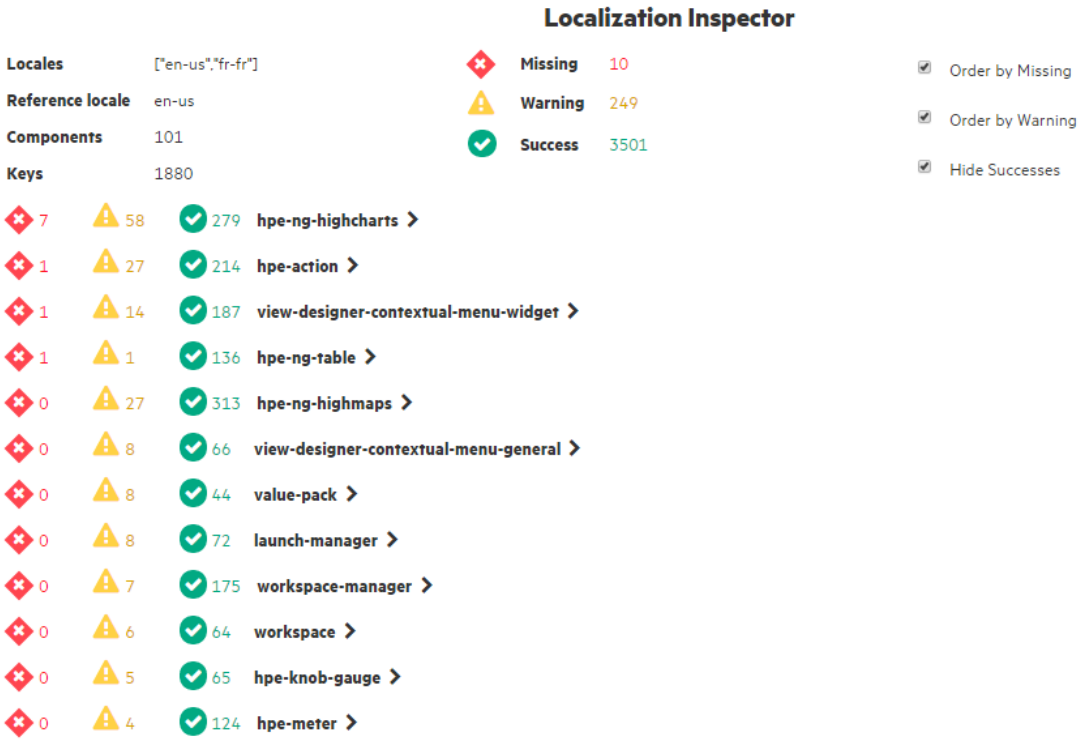


Figure 100 – Widget Localization Inspector

Chapter 20 Plugin Simulator

20.1 Overview

A generic plugin simulator is available by default. It can be used by a large audience for many different use cases:

- Add-on developers or integrators that prototype a Proof of Concept (Poc) or refine a demo for a customer in case no data server is available.
- Easy way to share demo and showcases without internet access.
- Prototype value packs (or packages) or integration of value packs cross domain.
- Help training to get very fast new kind of heterogeneous data
- Help developer to test their widgets, plugins… with different error and check the behavior of UOC…

The Plugin Simulator uses existing packages (based on the Unified Format), and is able to simulate data for dimensions, facts or object types (CRUD operation).

It can simulate realistic filtering, sorting, and granularity (hourly, daily, weekly…), date range, aggregation… and apply the RBAC model.

It can use pre-defined files to play (JSON or CSV format) or generate set of data with defined rules.

By default, the Plugin Simulator brings several simulated value packs (or packages) like Mobile Broad Band QoE, Catalog of objects and its CRUD operations…

NOTE: By Default, the Plugin Simulator is installed but disabled. You need to manually change its setting to make it enabled, and then be able to import its samples (packages, workspaces and views)

A plugin can have a plugin configuration file where you can turn on/off the plugin Simulator (property “active” that can be set to true or false)

Check <install_data_dir>/server/public/addons/plugins/plugin_simulator/config.json

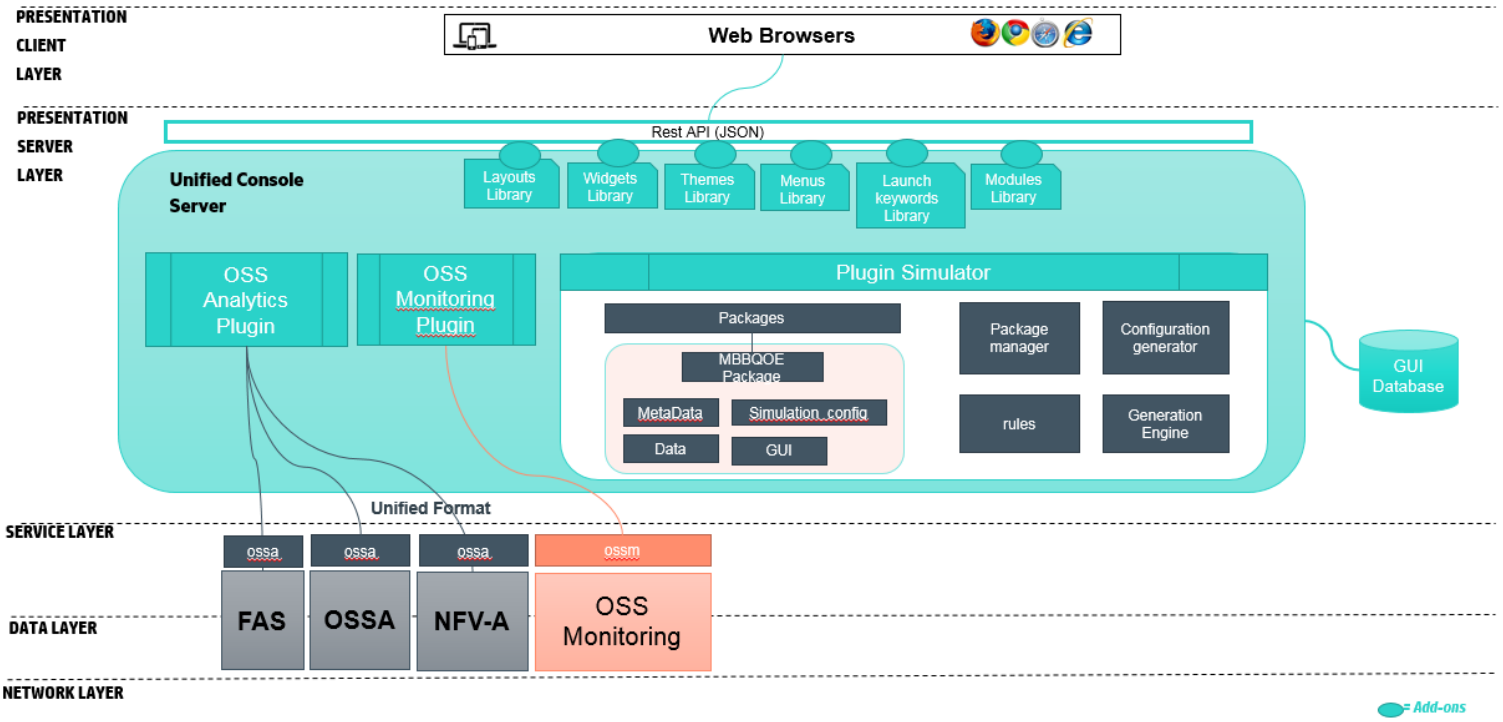


Figure 101 – Plugin Simulator Overview

As you can see on the figure above, the main components of the plugin simulator are:

- Packages
- Package manager
- Generation engine
- Rules

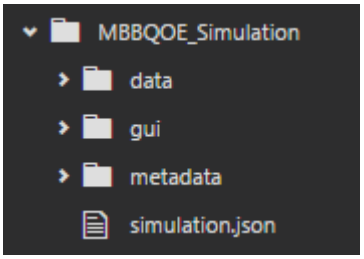
Note: the Configuration generator is not implemented yet

Packages is where you can add the simulation packages. They are located in **server/public/addons/plugins/plugin_simulator/packages/**

The other components (package manager, generation engine and rules) are located in **server/addons/plugins/plugin_simulator/** and you should not bring any modifications on them.

20.2 Packages

The simulation package follow the same architecture of a plugin's package. You can find **metadata**, **data** and a **GUI**. However, It must have a **simulation.json** file that allow user to configure the simulation among this package.



20.3 Simulation file

The simulation.json file allow configuration of the simulation among the package. It is described this way:

- **packageId**: package id
- **version**: version
- **description**: description of the package
- **dimensions**: To run a simulation over dimensions, we need to specify the values of each dimension
- **ruleRelations**: This is an array of simulation configurations. A configuration describes how to simulate data for a set of dimension(s) and a set of fact(s).
- **objects**: Object. Used to simulate object operations (CRUD or custom operations)

Here is a detailed description of the simulation.json file :

NOTE: To ease comprehension, the **ruleRelations** property is described below in 20.3.1 and **objects** property is described below in 20.3.2

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "title": "simulation.json",
  "description": "Description of the simulation.json file",
  "properties": {
    "packageId": {
```

```
      "type": "string",
      "title": "PackageId",
      "description": "Package unique identifier"
    },
    "version": {
      "type": "string",
      "title": "Version",
      "description": "Version of the package"
    },
    "description": {
      "type": "string",
      "title": "Description",
      "description": "Description of the package"
    },
    "dimensions": {
      "type": "array",
      "title": "Dimensions",
      "description": "It is needed to specify the possible dimension
values associated to a dimension",
      "items": {
        "type": "object",
        "title": "Dimensions id and values",
        "description": "Each item is a couple of a dimension id and a
definition of its values. Its values can be defined in a file (use
fileName property) or it can be defined in an array (use values
property)",
        "properties": {
          "id": {
            "type": "string",
            "title": "Dimension id",
            "description": "The dimension id (described in the package
metadata). You must specify the possible values of this dimension below"
          },
          "values": {
            "type": "array",
            "title": "Values",
            "description": "This option is not compatible with fileName.
Possible values for the dimension id mentioned above.",
            "items": {
                "type": "string",
                "title": "Value",
                "description": "Possible value the dimension can have"
            }
```

```
          },
          "fileName": {
              "type": "string",
              "title": "FileName",
              "description": "This option is not compatible with values.
If the dimension values are located in a data file located in
data/dimensions/json or data/dimensions/csv, you can specify the
fileName. It will get all the different values under the <dimensionId>
column "
          },
          "delimiter" : {
              "type": "string",
              "title": "Delimiter for csv files",
              "description": "CSV file delimiter",
              "default": ";"
          }
        },
        "required": [
          "id",
          "fileName",
          "values"
        ]
      }
    },
    "ruleRelations": {
        "type": "array",
        "title": "ruleRelations",
        "description": "Array of simulation rules. Each item is a
configuration for a simulation of a set of dimension(s) and a set of
fact(s)",
        "items": {
            "$ref": "#ruleRelation"
        }
    },
    "objects": {
        "type": "object",
        "title": "objects",
        "description": " Used to simulate object operations (CRUD or
custom operations)  "
        "properties": {
            "$ref": "#objects"
        }
    }
  },
```

```
  "required": [
    "packageId",
    "version",
    "description"
  ]
}
```

Here is an example of the first part of the simulation.json file:

```
{
  "packageId": "MBBQOE_Simulation",
  "version": "1.0",
  "description": "Simulation configuration for the MBBQOE simulation package",
  "dimensions": [{
    "id": "TECHNOLOGY",
    "fileName": "data_sample.csv",
    "delimiter": ","
  }, {
    "id": "BRAND",
    "values": ["Motorola", "Apple", "Samsung"]
  }],
  "ruleRelations": [...],
  "objects": {...}
}
```

20.3.1 ruleRelations

Rule relations define a simulation configuration for a set of dimension(s) and a set of fact(s). It is called ruleRelations because the simulation is based on a rule. For now, the plug-in simulator manage two rules: the **file rule** and the **random rule.** These ruleRelations have the same format but don't have the same parameters.

20.3.1.1 Simulation from a file

The file rule allow a simulation from a file defined in **data/dimFacts/csv/** or **data/dimFacts/json/.** Here is the definition of the ruleRelation for the file rule:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "title": "ruleRelation with the file rule",
  "description": "This ruleRelation use the file rule to generate data",
  "properties": {
    "dimensions": {
      "type": "array",
      "title": "Dimensions ids",
      "description": "The set of dimension ids you want to do a
simulation on",
      "items": {
        "type": "string",
```

```
        "title": "dimension id",
        "description": "a dimension id that will be used for the
simulation"
      }
    },
    "facts": {
      "type": "array",
      "title": "Facts ids",
      "description": "The set of fact ids you want to do a simulation
on",
      "items": {
        "type": "string",
        "title": "fact id",
        "description": "a fact id that will be used for the simulation"
      }
    },
    "rule": {
      "type": "object",
      "title": "Rule",
      "description": "The rule describes how to generate data for the
dimensions/facts relation",
      "properties": {
        "id": {
          "type": "string",
          "title": "Id",
          "description": "This is the id of the rule: which rule will be
used to generate data. the simulator manage two rules: 'random' or
'file'"
        },
        "params": {
          "type": "object",
          "title": "Params",
          "description": "Parameters that will configure the data
generation",
          "properties": {
              "path": {
                "type": "string",
                "title": "Path",
                "description": "The name of the data file (located in
/data/dimFacts/csv or /json) with its extension for instance
'data_sample.csv' or 'data_sample.json'"
              },
              "delimiter": {
                "type": "string",
```

```
                "title": "Delimiter",
                "description": "If the data file is in csv format, you
might need to specify the delimiter of the latest.",
                "default": ";"
              },
            "timestampColumn": {
              "type": "string",
              "title": "TimestampColumn",
              "description": "Is used to define the timestamp column
name inside the file, do not use this property if the data file does not
contain a timestamp column, do not use it if the user do not want to
have timestamp column in output"
            },
            "timestampDateFormat": {
              "type": "string",
              "title": "TimestampDateFormat",
              "description": "If your timestamp column is not in a unix
format, you must specify a format of the timestamp. for instance
YYYYMMDDHHmmss"
            },
            "manageAllSubset": {
              "type": "boolean",
              "title": "ManageAllSubset",
              "description": "When set to true, it will manage all the
subset of the dimensions/facts relation. For instance, when you have a
relation with 'dimA', 'dimB' and 'factA', 'factB'. The simulator will
generate data for every subset of this relation for instance : 'dimA'
and 'factB'",
              "default": false
            },
            "granularity": {
              "type": "integer",
              "title": "Granularity",
              "description": "You can specify the granularity of your
file. if you would like to use fileA when the end user is asking for a
900 granularity OR a fileB when he asks for a 3600 granularity, you
should define two ruleRelations with the same dim/fact relation but that
differs from the granularity. The simulator will pick the right
ruleRelation given the requested granularity"
            },
            "replicationType": {
              "type": "string",
              "title": "ReplicationType",
              "description": "The same as the granularity property. if
you would like to use fileA when the end user is asking for a day of
data OR a fileB when he asks for a week of data, you should define two
ruleRelations with the same dim/fact relation but that differs from the
```

```
replicationType. The simulator will pick the right ruleRelation given
the requested time range"
            },
            "replication": {
              "type": "object",
              "title": "Replication",
              "description": "it is an object that define if the
simulator will apply a replication on the data. If the replication is
set, the user will always have data, regardless of the requested time
range. If not set, the user must request data with a time range that
include time range defined in the data file",
              "properties": {
                "replication": {
                  "type": "boolean",
                  "title": "Replication",
                  "description": "Define if the replication will be
applied for the simulation",
                  "default": false
                },
                "replicationType": {
                  "type": "string",
                  "title": "ReplicationType",
                  "description": "'hourly', 'daily', 'weekly', 'monthly'
, 'yearly' define how the simulator will replicate the data from the
data file"
                }
              },
              "required": [
                "replication",
                "replicationType"
              ]
            },
            "aggregation": {
              "type": "object",
              "title": "Aggregation",
              "description": "Define how the data is aggregated",
              "properties": {
                "aggregationFunction": {
                  "type": "string",
                  "title": "AggregationFunction",
                  "description": "'SUM', 'AVG', 'MIN', 'MAX', it is a
mandatory string that define the aggregation function used to aggregate
the data"
                },
                "numberOfDigits": {
```

```
                  "type": "integer",
                  "title": "NumberOfDigits",
                  "description": "The number of digits you want to round
up the data aggregation",
                  "default": 2
                },
                "noAggreg": {
                    "type": "boolean",
                    "title": "noAggreg",
                    "description": "If you want to force no aggregation.
It will take the first occurrence of each dimension/fact relation"
                }
              },
              "required": [
                "aggregationFunction"
              ]
            }
          },
          "required": [
            "path",
            "aggregation"
          ]
        }
      },
      "required": [
        "id",
        "params"
      ]
    }
  },
  "required": [
    "facts",
    "rule"
  ]
}
```

20.3.1.2 Simulation from a file example

Here is an example of 3 ruleRelations using the file rule.

```
{
  "ruleRelations": [{
    "dimensions": ["TECHNOLOGY"],
    "facts": ["volume_up_sum"],
    "rule": {
      "id": "file",
      "params": {
        "timestampColumn": "EVENT_TIMESTAMP",
        "path": "data_sample.csv",
        "delimiter": ",",
        "granularity": 900,
        "replicationType": "daily",
        "replication": {
          "replication": true,
          "replicationType": "daily"
        },
        "aggregation": {
          "aggregationFunction": "SUM"
        }
      }
    }
  }, {
    "dimensions": ["TECHNOLOGY"],
    "facts": ["volume_up_sum"],
    "rule": {
      "id": "file",
      "params": {
        "timestampColumn": "EVENT_TIMESTAMP",
        "path": "data_sample.json",
        "granularity": 3600,
        "replication": {
          "replication": true,
          "replicationType": "daily"
        },
        "aggregation": {
          "aggregationFunction": "AVG",
          "noAggreg": false
        }
      }
    }
  }, {
    "dimensions": ["TECHNOLOGY", "BRAND"],
    "facts": ["volume_down_sum", "volume_up_sum"],
    "rule": {
      "id": "file",
      "params": {
        "path": "data_sample.json",
        "manageAllSubset": true,
        "replication": {
          "replication": true,
          "replicationType": "daily"
        },
        "aggregation": {
          "aggregationFunction": "MIN"
        }
      }
    }
```

```
  }]
}
```

20.3.1.3 Random simulation

The random simulation has quite the same format as a file rule simulation. Here is the definition of the random rule simulation :

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "title": "ruleRelation with the random rule",
  "description": "This ruleRelation use the random rule to generate
data",
  "properties": {
    "dimensions": {
      "type": "array",
      "title": "Dimensions ids",
      "description": "The set of dimension ids you want to do a
simulation on",
      "items": {
        "type": "string",
        "title": "dimension id",
        "description": "a dimension id that will be used for the
simulation"
      }
    },
    "facts": {
      "type": "array",
      "title": "Facts ids",
      "description": "The set of fact ids you want to do a simulation
on",
      "items": {
        "type": "string",
        "title": "fact id",
        "description": "a fact id that will be used for the simulation"
      }
    },
    "rule": {
      "type": "object",
      "title": "Rule ",
      "description": "The rule describe how to generate data for the
dimensions/facts relation",
      "properties": {
```

```
        "id": {
          "type": "string",
          "title": "Id ",
          "description": "This is the id of the rule: which rule will be
used to generate data. the simulator manage two rules: 'random' or
'file'"
        },
        "params": {
          "type": "object",
          "title": "Params ",
          "description": "Parameters that will configure the data
generation",
          "properties": {
              "manageAllSubset": {
                 "type": "boolean",
                 "title": "ManageAllSubset",
                 "description": "When set to true, it will manage all the
subset of the dimensions/facts relation. For instance, when you have a
relation with 'dimA', 'dimB' and 'factA', 'factB'. The simulator will
generate data for every subset of this relation for instance : 'dimA'
and 'factB'",
                 "default": false
              },
              "replication": {
               "type": "object",
               "title": "Replication",
               "description": "it is an object that define if the
simulator will apply a replication on the data",
               "properties": {
                 "replication": {
                   "type": "boolean",
                   "title": "Replication",
                   "description": "Define if the replication will be
applied for the simulation",
                   "default": false
                 },
                 "replicationType": {
                   "type": "string",
                   "title": "ReplicationType",
                   "description": "'hourly', 'daily', 'weekly', 'monthly'
, 'yearly' define how the simulator will replicate the randomized set of
data. The replication value should be higher that the periodicity value
set in the random property"
                 }
              },
```

```
            "required": [
              "replication",
              "replicationType"
            ]
          },
            "aggregation": {
              "type": "object",
              "title": "Aggregation",
              "description": "Define how the data is aggregated",
              "properties": {
                "aggregationFunction": {
                  "type": "string",
                  "title": "AggregationFunction",
                  "description": "'SUM', 'AVG', 'MIN', 'MAX', it is a
mandatory string that define the aggregation function used to aggregate
the data"
                },
                "numberOfDigits": {
                  "type": "integer",
                  "title": "NumberOfDigits",
                  "description": "The number of digits you want to
round up the data aggregation",
                  "default": 2
                },
                "noAggreg": {
                    "type": "boolean",
                    "title": "noAggreg",
                    "description": "If you want to force no
aggragtion. It will take the first occurence of each dimension/fact
relation"
                }
              },
              "required": [
                "aggregationFunction"
              ]
            },
          "granularity": {
            "type": "integer",
            "title": "Granularity",
            "description": "if you would like to use ruleRelationA
when the end user is asking for a 900 granularity OR ruleRelationB when
he asks for a 3600 granularity, you should define two ruleRelations with
the same dim/fact relation but that differs from the granularity. The
simulator will pick the right ruleRelation given the requested
granularity"
```

```
            },
            "random": {
              "type": "object",
              "title": "Random",
              "description": "This object defines the configuration of
the randomization",
              "properties": {
                  "periodicity": {
                    "type": "string",
                    "title": "Periodicity ",
                    "description": "'daily', 'weekly', 'monthly',
'yearly' used to define the time period of the generated data,  if the
user choose daily, it means that one day of data is generated"
                  },
                  "granularity": {
                    "type": "integer",
                    "title": "Granularity",
                    "description": "Defines the granularity of the
generated data. example: 900 will generate a point every 15 minutes",
                    "default": "3600"
                  },
                "commonRule": {
                  "type": "object",
                  "title": "CommonRule",
                  "description": "The commonRule defines the parameters
of the randomization applied to all generated data",
                  "properties": {
                    "minMax": {
                      "type": "array",
                      "title": "MinMax",
                      "description": "Array of two element, first
element present min value (min limit) and the second present the max
value(max limit). all generated data should be inside this interval.",
                      "items": {
                        "type": "integer",
                        "title": "Min and max limits",
                        "description": "Min and max limits"
                      }
                    },
                    "deltaMinMax": {
                      "type": "array",
                      "title": "DeltaMinMax ",
                      "description": "an array of two element, first
element is min value and the second element is the max value. It
```

```
represents the variation of the curve (ascending or descending). This
interval is used to generate the next point in the curve. for instance,
[10,20] will generate the next value of the curve that way:
previousValue + random value between 10 and 20. That way you can manage
the curves directions",
                      "items": {
                        "type": "integer",
                        "title": "Min and max interval",
                        "description": "Min and max interval"
                      }
                    },
                    "initValue": {
                      "type": "integer",
                      "title": "InitValue",
                      "description": "This is the starting point of the
values"
                    }
                  },
                  "required": [
                    "minMax",
                    "deltaMinMax",
                    "initValue"
                  ]
                },
                "singleRule": {
                  "type": "array",
                  "title": "SingleRule",
                  "description": "array of objects, it defines the list
of single rule to be applied to a specific fact column. that way you can
have different randomization for different facts",
                  "items": {
                    "type": "object",
                    "title": "singleRule item",
                    "description": "a singleRule item has the same
properties as the common rule. It specifies a fact and (not mandatory)
dimValues",
                    "properties": {
                      "fact": {
                        "type": "string",
                        "title": "Fact id",
                        "description": "The fact id you want to apply
the randomization defined below"
                      },
                      "dimValues": {
                        "type": "array",
```

```
                        "title": "DimValues",
                        "description": "When dimension values are
specified, the randomization defined below will be applied to these
dimvalues/fact relation only. The others dimValues/fact relations will
follow the commonRule",
                        "items": {
                          "type": "string",
                          "title": "DimValue",
                          "description": "Dimension value"
                        }
                      },
                      "minMax": {
                        "type": "array",
                        "title": "MinMax",
                        "description": "Array of two element, first
element present min value (min limit) and the second present the max
value(max limit). all generated data should be inside this interval.",
                        "items": {
                          "type": "integer",
                          "title": "Min and max limits",
                          "description": "Min and max limits"
                        }
                      },
                      "deltaMinMax": {
                        "type": "array",
                        "title": "DeltaMinMax ",
                        "description": "an array of two element, first
element is min value and the second element is the max value. It
represents the variation of the curve (ascending or descending). This
interval is used to generate the next point in the curve. for instance,
[10,20] will generate the next value of the curve that way:
previousValue + random value between 10 and 20. That way you can manage
the curves directions",
                        "items": {
                          "type": "integer",
                          "title": "Min and max interval",
                          "description": "Min and max interval"
                        }
                      },
                      "initValue": {
                        "type": "integer",
                        "title": "InitValue",
                        "description": "This is the starting point of
the values",
                        "default": 0
```

```
                    }
                  },
                  "required": [
                    "fact",
                    "minMax",
                    "deltaMinMax",
                    "initValue"
                  ]
                }
              }
            },
            "required": [
              "commonRule",
              "periodicity",
              "singleRule"
            ]
          }
        },
        "required": [
          "aggregation",
          "granularity",
          "random"
        ]
      }
    },
    "required": [
      "id",
      "params"
    ]
  }
},
"required": [
  "dimensions",
  "facts",
  "rule"
]
}
```

20.3.1.4 Random simulation example

Here is an example of 2 ruleRelations using the random rule

```
{
```

```
"ruleRelations": [{
  "dimensions": ["TECHNOLOGY", "BRAND"],
  "facts": ["volume_up_sum"],
  "rule": {
    "id": "random",
    "params": {
      "aggregation": {
        "aggregationFunction": "SUM"
      },
      "granularity": 900,
      "random": {
        "commonRule": {
          "minMax": [10, 500],
          "deltaMinMax": [0, 30],
          "initValue": 30
        },
        "periodicity": "weekly",
        "singleRule": [{
          "fact": "volume_up_sum",
          "dimValues": ["3G", "Motorola"],
          "minMax": [0, 500],
          "deltaMinMax": [-40, 0],
          "initValue": 500
        }, {
          "fact": "volume_up_sum",
          "dimValues": ["4G", "Apple"],
          "minMax": [0, 500],
          "deltaMinMax": [5, 10],
          "initValue": 200
        }, {
          "fact": "volume_up_sum",
          "dimValues": ["4G", "Samsung"],
          "minMax": [0, 500],
          "deltaMinMax": [0, 30],
          "initValue": 0
        }]
      }
    }
  }
}, {
  "dimensions": ["TECHNOLOGY", "BRAND"],
  "facts": ["volume_down_sum", "volume_up_sum"],
  "rule": {
    "id": "random",
    "params": {
      "aggregation": {
        "aggregationFunction": "AVG"
      },
      "replication": {
        "replication": true,
        "replicationType": "monthly"
      },
      "random": {
        "commonRule": {
          "minMax": [10, 3000],
          "deltaMinMax": [0, 30],
          "initValue": 100
```

```
          },
          "minMax": [10, 100],
          "periodicity": "weekly",
          "singleRule": [{
            "fact": "volume_up_sum",
            "dimValues": ["2G", "Motorola"],
            "minMax": [10, 500],
            "deltaMinMax": [0, 10],
            "initValue": 200
          }, {
            "fact": "volume_down_sum",
            "dimValues": ["4G", "Samsung"],
            "minMax": [0, 6000],
            "deltaMinMax": [0, -60],
            "initValue": 6000
          }, {
            "fact": "volume_down_sum",
            "dimValues": ["4G", "Apple"],
            "minMax": [0, 4000],
            "deltaMinMax": [0, -40],
            "initValue": 4000
          }]
        }
      }
    }
  }]
}
```

20.3.2 Objects / Operations simulation

The object simulation allow a simulation upon object operation. For instance for a certain object id, when the user use the operation update, you can use the object simulation to simulate a notification. Here is the definition of the object simulation object.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "title": "Object simulation",
  "description": "Allow a simulation upon object operation",
  "properties": {
    "objects": {
      "type": "object",
      "title": "Objects",
      "description": "Used to simulate object operations (CRUD or custom
operations)",
      "properties": {
        "<OBJECT_ID>": {
          "type": "object",
          "title": "<OBJECT_ID>",
          "description": "The Id of the object on which we want to
simulate operations",
```

```
          "properties": {
            "operations": {
              "type": "array",
              "title": "Operations",
              "description": "Array of operations",
              "items": {
                "type": "object",
                "title": "Operation",
                "description": "This is where we are going to simulate
the action for this operation",
                "properties": {
                  "<OPERATION_ID>": {
                    "type": "array",
                    "title": "<OPERATION_ID>",
                    "description": "Id of the object's operation on
which we are going to simulate",
                    "items": {
                      "type": "object",
                      "title": "A simulation for the operation ID",
                      "description": "A set of 4 properties: conditions,
outputs, response and asyncResponse. You can have different simulations
for different conditions for a same operation",
                      "properties": {
                        "conditions": {
                          "type": "array",
                          "title": "Conditions",
                          "description": "A set of conditions related to
this simulation",
                          "items": {
                            "type": "object",
                            "title": "Condition",
                            "description": "A condition that limit the
use of this simulation. If the condition is met, this simulation will be
used",
                            "properties": {
                              "id": {
                                "type": "string",
                                "title": "Id",
                                "description": "Id of the object
property the condition rely on"
                              },
                              "<FILTER_ID>": {
                                "type": "array",
                                "title": "<FILTER_ID>",
```

```
                                "description": "Possible FILTER_ID
operand are: 'eq', 'not_eq', 'startsWith', 'not_startsWith', 'contains',
'not_contains', 'endsWith', 'not_endsWith', 'gt', 'not_gt', 'ge',
'not_ge', 'lt', 'not_lt', 'le', 'not_le', 'isnull', 'notnull'",
                                "items": {
                                  "type": [
                                    "string",
                                    "number"
                                  ],
                                  "title": "Value of the filter related
to the operand",
                                  "description": "Value of the filter
related to the operand. It can be '*' to accept everything"
                                }
                              }
                            },
                            "required": [
                              "id",
                              "<FILTER_ID>"
                            ]
                          }
                        },
                        "outputs": {
                          "type": "array",
                          "title": "Outputs",
                          "description": "Modifications you want to
apply to the object when the condition is met",
                          "items": {
                            "type": "object",
                            "title": "Modification",
                            "description": "Modification that will be
applied to the object",
                            "properties": {
                              "id": {
                                "type": "string",
                                "title": "Id",
                                "description": "Id of the object
property the modification will be on"
                              },
                              "value": {
                                "type": "string",
                                "title": "Value",
                                "description": "Value that the property
<id> will take after the operation"
```

```
                    }
                  },
                  "required": [
                    "id",
                    "value"
                  ]
                }
              },
              "response": {
                "type": "object",
                "title": "Response",
                "description": "Http server response",
                "properties": {
                  "statusCode": {
                    "type": "integer",
                    "title": "StatusCode",
                    "description": "Status code of the http
response"
                  },
                  "notification": {
                    "$ref": "#notification"
                  },
                  "duration": {
                    "type": "string",
                    "title": "Duration",
                    "description": "Moment value. Time the
simulator will take before returning the response after the action"
                  }
                },
                "required": [
                  "statusCode",
                  "notification"
                ]
              },
              "asyncResponse": {
                "type": "object",
                "title": "AsyncResponse",
                "description": "Used when you want to simulate
an asynchronous response",
                "properties": {
                  "duration": {
                    "type": "string",
```

```
                              "title": "Duration",
                              "description": "Moment value. Time the
simulator will take before returning the response after the action"
                            },
                            "notification": {
                              "$ref": "#notification-composite-key"
                            }
                          },
                          "required": [
                            "duration",
                            "notification"
                          ]
                        }
                      },
                      "required": [
                        "conditions",
                        "outputs",
                        "response"
                      ]
                    }
                  }
                }
              }
            },
            "required": [
              "operations"
            ]
          }
        }
      }
    },
    "required": [
      "objects"
    ]
}
```

Example of an objects / operations simulation

```
{
  "objects": {
    "viewCatalogItem": {
      "operations": [{
        "create": [],
```

```
        "deliver": [{
          "conditions": [{
            "id": "name",
            "eq": "*"
          }],
          "outputs": [{
            "id": "status",
            "value": "Available"
          }],
          "response": {
            "statusCode": 200,
            "notification": {
              "type": "alert",
              "level": "success",
              "id": "operation_viewCatalogItem_delivred",
              "data": {
                "title": {
                  "en-us": "Operation successfull",
                  "fr-fr": "Opération réussi"
                },
                "text": {
                  "en-
us": "Operation successfull with operation \"Deliver\" on object of type \"viewCatalogItem\"",
                  "fr-
fr": "Opération réussi pour l'opération \"Deliver\" sur l'objet de type \"viewCatalogItem\""
                }
              },
              "display": {
                "type": "popup"
              }
            },
            "duration": "2s"
          },
          "asyncResponse": {
            "duration": "4s",
            "notification": {
              "type": "alert",
              "id": "catalog_item-delivered",
              "origin": "plugin",
              "domain": "plugin_simulator",
              "package": "Healthcare_Simulation"
            }
          }
        }]
      }]
    },
    "serviceReadinessInstance": {
      "operations": [{
        "start": [{
          "conditions": [{
            "id": "serviceStatus",
            "eq": "Ready"
          }],
          "outputs": [{
            "id": "serviceStatus",
            "value": "In Service"
          }],
```

```
    "response": {
      "statusCode": 200,
      "notification": {
        "type": "alert",
        "level": "success",
        "id": "operation_serviceReadinessInstance_started",
        "data": {
          "title": {
            "en-us": "Operation successfully started",
            "fr-fr": "Opération démarrée avec succès"
          },
          "text": {
            "en-us": "The service has been successfully started",
            "fr-fr": "Le service a été démarré avec succès"
          }
        },
        "display": {
          "type": "toast",
          "options": {
            "timeout": 5000
          }
        }
      },
      "duration": "1s"
    }
  }, {
    "conditions": [{
      "id": "serviceStatus",
      "eq": "In Service"
    }],
    "response": {
      "statusCode": 409,
      "notification": {
        "type": "alert",
        "level": "warn",
        "id": "operation_serviceReadinessInstance_starting-error",
        "data": {
          "title": {
            "en-us": "Operation aborted - Starting error",
          "fr-fr": "Opération annulée - Erreur de démarrage"
        },
        "text": {
           "en-us": "The service is already started",
           "fr-fr": "Le service est déjà démarré"
        }
      }
    },
    "duration": "1s"
  },
  "asyncResponse": {
    "duration": "4s",
    "notification": {
      "type": "alert",
      "id": "operation_readiness-already_started",
      "origin": "plugin",
      "domain": "plugin_simulator",
      "package": "Healthcare_Simulation"
```

```
            }
          }
        }]
      }]
    }
}
}
```

Chapter 21 Security Tools

Security is very important for Unified Console, it is also up to each Add-on developer to grant a high level of security. To ease the selection of open sources in your add-ons and make sure there is no vulnerabilities detected (and possibly fixed in a next version), it is strongly recommended to integrate in your delivery step, a complete checking using these tools.

- Node Security Platform (NSP)
- David

Other tools exists and can be easily integrated in grunt task in needed.

21.1 Checking vulnerable Node modules

You need to install the tool “name Security Project” (nsp) commandline tool. You can see details here: https://github.com/nodesecurity/nsp

For more information on Node.js security, check out the nodesecurity.io Resources page.

21.1.1 Install Node Security Platform (NSP) tool

To install the **nsp** package, type the following command into your Terminal:

```
$ [sudo] npm install nsp -g
```

21.1.2 Run the audit

Once the `nsp` module is installed globally, you can check for vulnerable modules in your project using one of the two following commands:

- **`nsp audit-package`**: Takes an existing *package.json* file and submits it for validation to nodesecurity.io.

The first step after finding vulnerable modules is to check for updates. Hopefully, updating to the latest version of a module will resolve the security warning

21.2 Checking outdated Node modules

A great command line tool for this is david (https://github.com/alanshaw/david), or you can use the built-in `$ npm outdated` command.

21.2.1 Install and use david

```
$ [sudo] npm install david -g
```

You can search for outdated modules with david by simply typing:

```
$ david
```

This will check your local package.json file for outdated modules and list newer versions.

21.2.2 Npm outdated

Another option for checking for outdated modules is npm's very own npm outdated command.

The `npm outdated` command has an optional `depth` flag which allows you to control how many levels deep to check for outdated dependencies. To only check the top level dependencies and devDependencies, you can specify a depth of 0:

```
$ npm outdated --depth 0

Package       Current  Wanted  Latest  Location

crumb           2.2.0   2.2.0   3.1.0  crumb

libyaml         0.2.2   0.2.2   0.2.4  libyaml

marked          0.3.0   0.3.0   0.3.2  marked

qs              0.6.6   0.6.6   1.2.2  qs

request        2.39.0  2.39.0  2.40.0  request

syntax-error    1.0.0   1.0.0   1.1.1  syntax-error
```

Similarly, to check for outdated global modules, you can pass the `--global` or `-g` flag:

```
$ npm outdated -g --depth 0

Package    Current    Wanted    Latest  Location

express     3.16.0     4.8.4     4.8.4  /usr/local/lib > express

npm         1.4.21    1.4.23    1.4.23  /usr/local/lib > npm
```

Chapter 22
Troubleshooting

22.1 Web Browser Console

If you have issues with the web browser, please check carefully that you are using a supported web navigator. HTML 5 is only supported in very last web browser versions.

There is no logging at the client side and no persistence of console messages. If the problem is persistent, it is recommended to help the support team by opening the web browser console and check any unexpected message (error, warning...). This console can be visible usually by pressing F12 (check your navigator documentation for details).

NOTE: If you notice unexpected behavior after changes done at the UOC Server, It is strongly recommended to clear the web browser cache and restart from a fresh web browser usage.

22.2 Troubleshooting UOC Servers

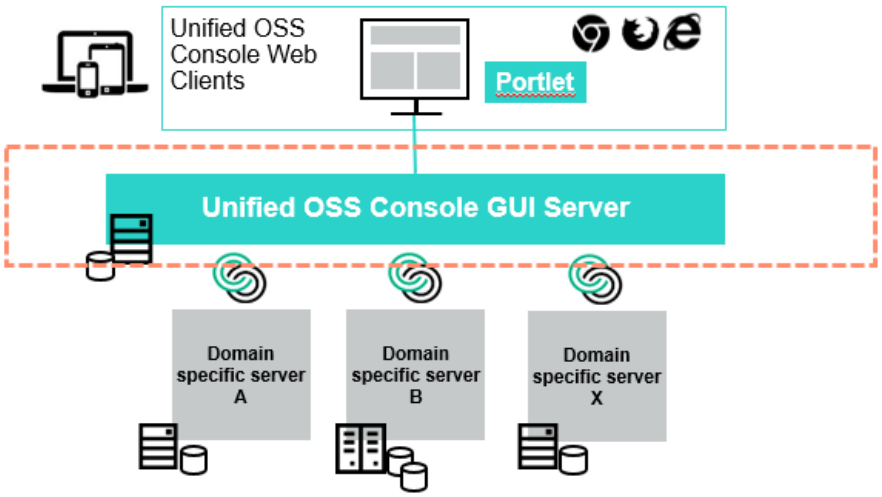


Figure 102: Troubleshooting UOC Server

Output Console displays useful information about the UOC server instance after the administrator start the platform:

```
$ uoc2 start
```

The following information are displayed:

- All RESTAPI dynamically loaded by categories (~core public interfaces)
- All RESTAPI dynamically loaded from installed Plugins (~plugins public interfaces)
- plugin name, version, host server, port server, demo (true|false)
- Listening Port (default is 3000)
- Listening protocol (default is http)
- Server timeout (default is unlimited)
- Node environment (dev or prod)
- UOC Node root
- Authentication (local or SAML)
- GUI Database information (server, port, protocol)
- Startup parameter (import policy)
- GUI database initialization logs
- All these info are logged in logs files.

NOTE: IF some plugins make the platform unstable, you can disable it at startup (turn off 'active' in configuration file in /server/public/addons/<pluginId>/conf

22.3 Troubleshooting UOC Plugins

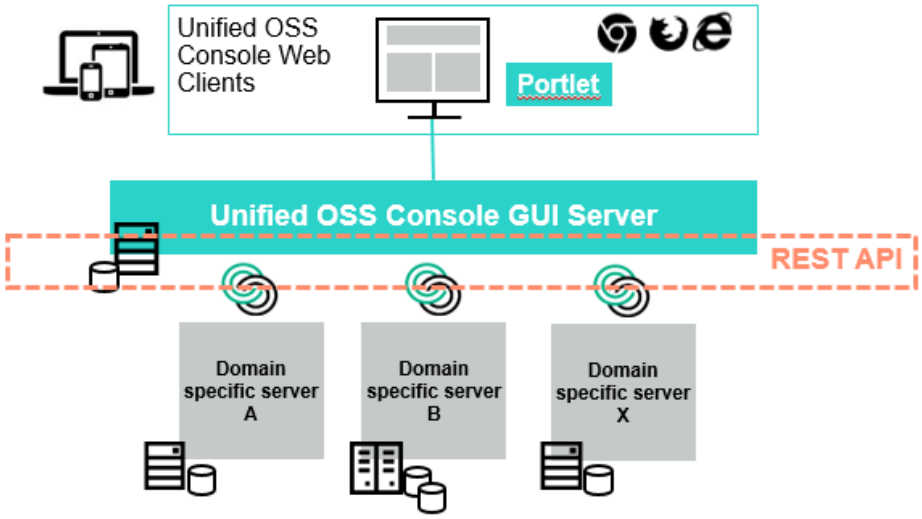


Figure 103: Troubleshooting UOC Plugins

You can check if a specific Plugin interface is working calling directly the public RESTAPI. This interface is used by the web browser to get information and generates GUI.

You can list the available packages for this plugins, and get a list of all packages.

Returns a list of packages for this plugin (ossa) in JSON format

http://localhost:3000/V1.0/domains/ossa/packages

You can get details of 1 package from the list.

http://localhost:3000/V1.0/domains/ossa/packages/MBBQOE_Trial

You can get list of all views for this plugin

http://localhost:3000/V1.0/domains/ossa/packages/views

You can get list of views of this package

http://localhost:3000/V1.0/domains/ossa/packages/MBBQOE_Trial/views

You can get list of all workspaces for this plugin

http://localhost:3000/V1.0/domains/ossa/packages/workspaces

You can get list of workspaces of this package

http://localhost:3000/V1.0/domains/ossa/packages/MBBQOE_Trial/workspaces

22.4 Troubleshooting UOC Widgets

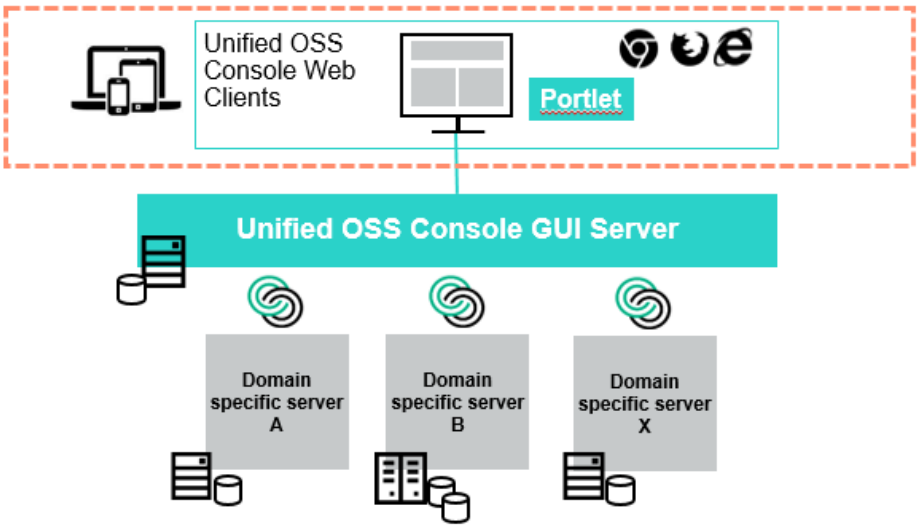


Figure 104: Troubleshooting UOC Widgets

All the client components can be troubleshoot with thee web browser in developer / debug mode (usually F12 key)

The Web Browser console let you display:

- Console logs of the web browser. (by severity)
- URL requested to the UOC Server (response, data…)
- Timeframe of response

The web browser also allow you to clear the cache if you think an issue is linked to a cache pb (css, data,…)

IT also allow you to set breakpoint and debug the code, BUT in the runtime kit, all the code has been minified (to reduce footprint), so it will not very easy without a sdk version to get details of the pb trying to debug the code.

➔ All the http logs and UOC object access are logged by the server in the file <installDir>/logs

Recommandation:

Support of web applications can be tricky, so you need to multiple testing to isolate the root cause.

- Test on several browsers
- Test on several Devices
- Test on several OS

It is strongly recommended to try several web browser if you see graphical issue or limitation. It could be linked to a specific version of OS, patch level from the web browser used.

It is strongly recommended to try on several device if you notice issue or graphical limitation on a specific device (ex: table). It could be linked to a responsive layout that is wrongly implemented and do not display the expected result.

22.5 Troubleshooting Data / Domain Servers

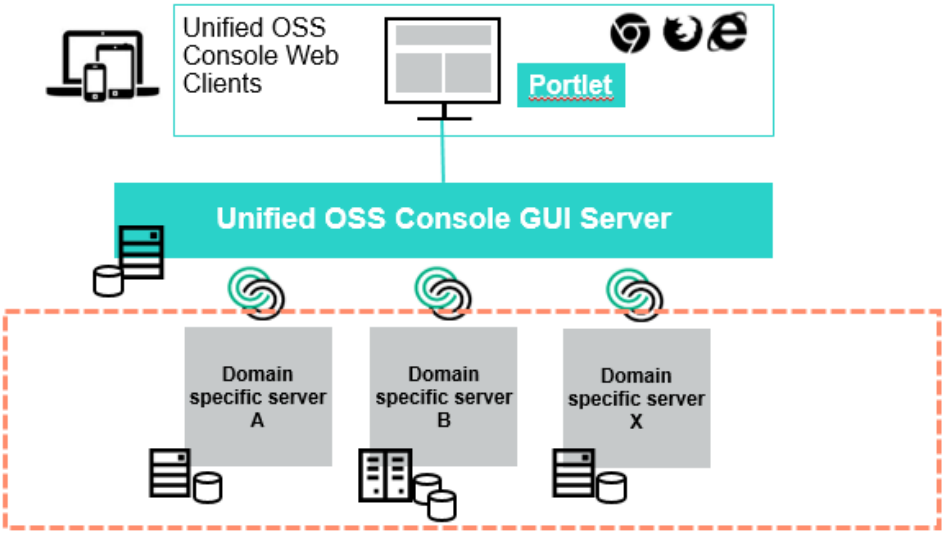


Figure 105: Troubleshooting Data / Domain Servers

To troubleshoot the Data server, you need to understand which server is queried and what is the protocol used. It is totally custom by data server and all the api are heterogeneous. You need to read carefully the documentation to see how troubleshoot such server.

Tips for OSSA Server (OSS Analytics Foundation)

For OSSA Server, you can easily call the REST API to get 'raw' data from the data server.

List available packages (where ossv031.gre.hp.com is an OSS Analytics server)

http://ossv031.gre.hp.com:8080/ossa/packages

Details of one specific packages:

http://ossv031.gre.hp.com:8080/ossa/packages/MBBQOE_Trial

You can also collect data with the right syntax. A cookbook exist from the OSSA server team that details all the features

All the request done by the plugin to the data or domain server are logged in the console (server side)

```
>>>url: http://ossv031.gre.hp.com:8080/ossa/packages/MBBQOE_Trial/facts/Web_quality/streaming_qualit
y/file_sharing_quality/email_quality/dims/IMSI/timewindow/1418842800000/1426672800000?granularity=15
&batchsize=1000000&wheredim=IMSI,eq,208331000000005&roles=Platform%20Administrator,User%20Administrat
or,Package%20Designer,Operator_L1,Operator_L2,Operator_L3,Authorized_Operator_For_Loc,Report_Exporte
r,Guest
OSSA-Sending response headers: [ 'EVENT_TIMESTAMP', 'BRAND', 'volume_up_sum' ] first record: [ 14240
74500000, 'Alcatel', 5206848 ] status: OK undefined
OSSA-Sending response headers: [ 'EVENT_TIMESTAMP',
  'IMSI',
  'web_score',
  'streaming_score',
  'file_sharing_score',
  'email_score' ] first record: [ 1424074500000, '208331000000005', null, null, null, null ] status:
 OK undefined
OSSA-Sending response headers: [ 'EVENT_TIMESTAMP',
  'IMSI',
  'Web_retainability',
  'streaming_retainability',
  'file_sharing_retainability',
  'email_retainability' ] first record: [ 1424074500000, '208331000000005', null, null, null, null ]
 status: OK undefined
OSSA-Sending response headers: [ 'EVENT_TIMESTAMP',
  'IMSI',
  'Web_quality',
  'streaming_quality',
  'file_sharing_quality',
  'email_quality' ] first record: [ 1424074500000, '208331000000005', null, null, null, null ] statu
s: OK undefined
```

Figure 106: Example of troubleshooting Data / Domain Servers (OSSA)

22.6 Troubleshooting GUI Database (Apache CouchDB)

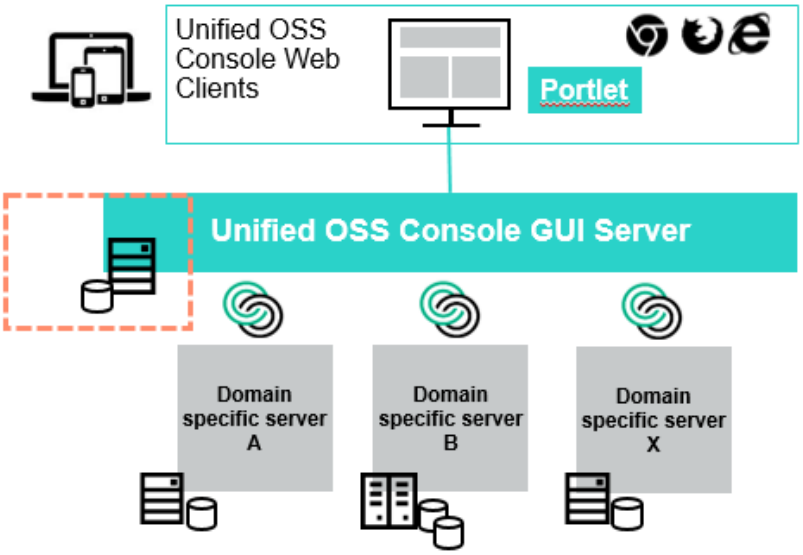


Figure 107: Troubleshooting GUI Database

CouchDB stores data as "documents", as one or more field/value pairs expressed as JSON. Field values can be simple things like strings, numbers, or dates; but ordered lists and associative arrays can also be used. Every document in a CouchDB database has a unique id and there is no required document schema.

http://couchdb.apache.org/

It is easy to use the administration tool to list and manage data stored in this database.

Check if the DB is alive: http://localhost:5984 return a JSON answer.

CouchDB includes a web-based front end called Futon which can be accessed by your browser via the following address: http://localhost:5984/_utils/

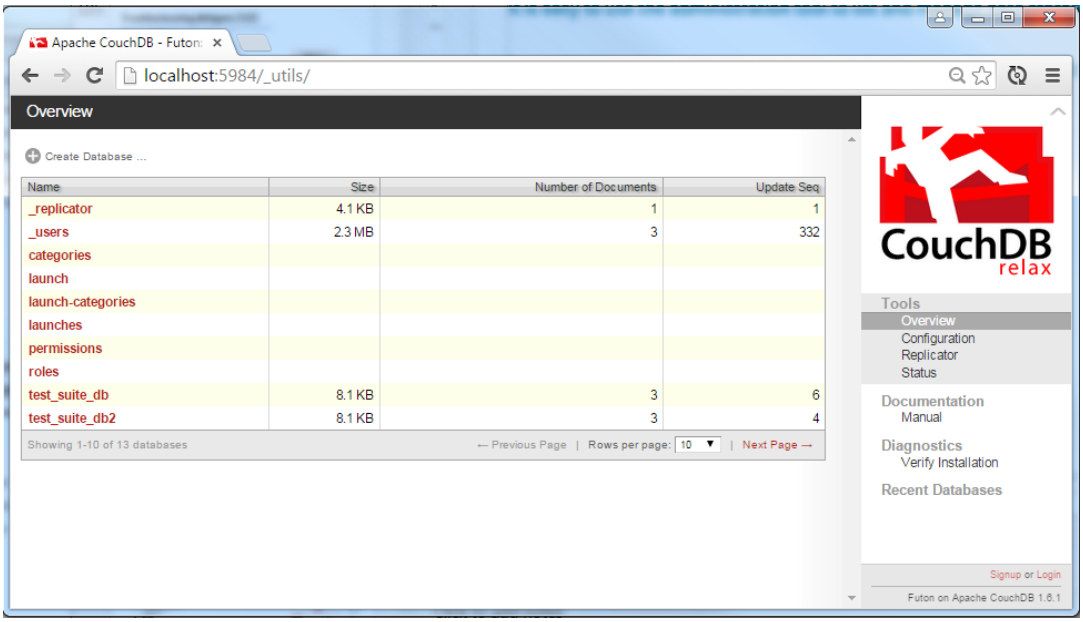


Figure 108: Apache CouchDB Front end Futon

Chapter 23
Open Sources Listing

UOC integrates many open sources. You can find in the following section, details on these open sources and their official web site to get more information or follow their features and fixes.

23.1 Server Open Sources

These open sources are integrated in the UOC server.

Runtime kit

Open Source	Version	License	Web Site
async	2.0.1	Copyright (c) 2010-2016 Caolan McMahon	https://github.com/caolan/async
body-parser	1.15.2	MIT license	https://github.com/expressjs/body-parser
compression	1.6.2	MIT license	https://github.com/expressjs/compression
cookie-parser	1.4.3	MIT license	https://github.com/expressjs/cookie-parser
cron	1.1.1	MIT license	https://github.com/ncb000gt/node-cron
csurf	1.9.0	MIT license	https://github.com/expressjs/csurf
csvtojson	1.0.0	Copyright (C) 2013 Keyang Xiang	https://github.com/Keyang/node-csvtojson
ejs	2.5.1	MIT license	http://embeddedjs.com/
express	4.14.0	MIT license	http://expressjs.com/
express-jwt	5.1.0	MIT license	https://github.com/auth0/express-jwt
express-validator	2.20.10	MIT license	https://github.com/ctavan/express-validator
frameguard	2.0.0	MIT license	https://github.com/helmetjs/frameguard
helmet	2.3.0	MIT license	https://github.com/helmetjs/helmet
helmet-csp	1.2.2	MIT license	https://github.com/helmetjs/csp
Hsts	1.0.0	MIT license	https://github.com/helmetjs/hsts
jsonschema	1.0.1	MIT license	https://github.com/tdegrunt/jsonschema
jsonwebtoken	7.1.9	MIT license	https://github.com/auth0/node-jsonwebtoken
lodash	3.10.1	Copyright 2012-2014 The Dojo Foundation	https://github.com/lodash/lodash
log4JS	0.6.38	Apache 2.0 License	https://github.com/nomiddlename/log4js-node
moment	2.15.1	Copyright (c) 2011-2016 Tim Wood, Iskren Chernev, Moment.js contributors	https://github.com/moment/moment
moment-duration-format	1.3.0	MIT license	https://github.com/jsmreese/moment-duration-format
ms	0.7.1	MIT license	https://github.com/rauchg/ms.js
Msgpack-js	0.3.0	MIT license	https://github.com/creationix/msgpack-js

nano	6.1.5	Apache 2.0 License	https://github.com/dscape/nano
nconf	0.8.4	Copyright (c) 2011 Nodejitsu Inc.	https://github.com/flatiron/nconf
passport	0.3.2	MIT license	http://passportjs.org/
passport-local	1.0.0	MIT license	https://github.com/jaredhanson/passport-local
passport-saml	0.15.0	MIT license	https://github.com/bergie/passport-saml
pdfkit	0.8.0	MIT license	https://github.com/devongovett/pdfkit
phantomjs	1.9.19	MIT license	https://github.com/ariya/phantomjs/
q	1.4.1	MIT license	https://github.com/kriskowal/q
Redis	2.6.2	BSD license	https://github.com/antirez/redis
request	2.75.0	Apache 2.0 License	https://github.com/mikeal/request
socket.io	1.5.0	MIT license	https://github.com/socketio/socket.io
socket.io-adapter	0.4.0	MIT license	https://github.com/socketio/socket.io-adapter
socketio-jwt	4.5.0	MIT license	https://github.com/auth0/socketio-jwt
Uid2	0.0.3	MIT license	https://github.com/coreh/uid2
xml2js	0.4.17	MIT license	https://github.com/Leonidas-from-XIV/node-xml2js
Yargs	6.2.0	MIT license (c)2010 James Halliday (mail@substack.net) Modified work (c)2014 Contributors (ben@npmjs.com)	https://github.com/yargs/yargs http://yargs.js.org/

Figure 109: Open Sources integrated in UOC V2.3 (Server)

SDK kit

Open Source	Version	License	Web Site
Chai	3.5.0	MIT license	https://github.com/chaijs/chai
Connect-livereload	0.6.0	Copyright (c) 2013 intesso	https://github.com/intesso/connect-livereload
Grunt	1.0.1	Copyright jQuery Foundation and other contributors, https://jquery.org/	https://github.com/gruntjs/grunt
Grunt-autoprefixer	3.0.4	MIT license	https://github.com/nDmitry/grunt-autoprefixer
Grunt-concurrent	2.3.1	MIT license	https://github.com/sindresorhus/grunt-concurrent
Grunt-contrib-clean	1.0.0	MIT license	https://github.com/gruntjs/grunt-contrib-clean
Grunt-contrib-compress	1.3.0	MIT license	https://github.com/gruntjs/grunt-contrib-compress
Grunt-contrib-copy	1.0.0	MIT license	https://github.com/gruntjs/grunt-contrib-copy

Grunt-contrib-csslint	2.0.0	MIT license	https://github.com/gruntjs/grunt-contrib-csslint
Grunt-contrib-cssmin	1.0.2	MIT license	https://github.com/gruntjs/grunt-contrib-cssmin
Grunt-contrib-htmlmin	2.0.0	MIT license	https://github.com/gruntjs/grunt-contrib-htmlmin
Grunt-contrib-jshint	1.0.0	MIT license	https://github.com/gruntjs/grunt-contrib-jshint
Grunt-contrib-uglify	2.0.0	MIT license	https://github.com/gruntjs/grunt-contrib-uglify
Grunt-contrib-watch	1.0.0	MIT license	https://github.com/gruntjs/grunt-contrib-watch
Grunt-contrib-yuidoc	1.0.0	MIT license	https://github.com/gruntjs/grunt-contrib-yuidoc
Grunt-env	0.4.4	Apache license 2.0	https://github.com/jsoverson/grunt-env
Grunt-express-server	0.5.3	MIT license	https://github.com/ericclemmons/grunt-express-server
Grunt-istanbul	0.7.1	MIT license	https://github.com/taichi/grunt-istanbul
Grunt-karma	2.0.0	Copyright (c) 2013 Dave Geddes	https://github.com/karma-runner/grunt-karma
Grunt-mocha-test	0.13.2	MIT license	https://github.com/pghalliday/grunt-mocha-test
Grunt-newer	1.2.0	MIT license	https://github.com/tschaub/grunt-newer
Grunt-ng-annotate	2.0.2	MIT license	https://github.com/mgol/grunt-ng-annotate
Grunt-nodemon	0.4.2	Copyright (c) 2013 Chris Wren	https://github.com/ChrisWren/grunt-nodemon
Grunt-open	0.2.3	Copyright (c) 2013 Jarrod Overson	https://github.com/jsoverson/grunt-open
Grunt-sass	1.2.1	MIT license	https://github.com/sindresorhus/grunt-sass
Grunt-shell	2.0.0	MIT license	https://github.com/sindresorhus/grunt-shell
Jshint-stylish	2.2.1	MIT license	https://github.com/sindresorhus/jshint-stylish
Karma	1.3.0	MIT license	https://github.com/karma-runner/karma
Karma-chrome-launcher	2.0.0	MIT license	https://github.com/karma-runner/karma-chrome-launcher
Karma-coverage	1.1.1	MIT license	https://github.com/karma-runner/karma-coverage
Karma-firefox-launcher	1.0.0	MIT license	https://github.com/karma-runner/karma-firefox-launcher
Karma-ie-launcher	1.0.0	MIT license	https://github.com/karma-runner/karma-ie-launcher
Karma-junit-reporter	1.1.0	MIT license	https://github.com/karma-runner/karma-junit-reporter
Karma-mocha	1.2.0	MIT license	https://github.com/karma-runner/karma-mocha
Karma-phantomjs-launcher	1.0.2	MIT license	https://github.com/karma-runner/karma-phantomjs-launcher
Karma-requirejs	1.1.0	MIT license	https://github.com/karma-runner/karma-requirejs
Karma-spec-reporter	0.0.26	Copyright (C) 2015 Michael Lex	https://github.com/mlex/karma-spec-reporter
Karma-tap-reporter	0.0.6	MIT license	https://github.com/fumiakiy/karma-tap-reporter
Load-grunt-tasks	3.5.2	MIT license	https://github.com/sindresorhus/load-grunt-tasks

Mocha	3.1.2	MIT license	https://github.com/mochajs/mocha
Requirejs	2.3.2	Copyright jQuery Foundation and other contributors	https://github.com/requirejs/requirejs
Time-grunt	1.4.0	MIT license	https://github.com/sindresorhus/time-grunt
Yeoman-environment	1.6.5	BSD license	https://github.com/yeoman/environment
Yeoman-generator	0.24.1	BSD license Copyright (c) Google	https://github.com/yeoman/generator
Yeoman-test	1.5.1	MIT license	https://github.com/yeoman/yeoman-test
Yeoman-assert	2.2.1	BSD license Copyright (c) Google	https://github.com/yeoman/yeoman-assert

23.2 Client Open Sources

These open sources are integrated in the UOC server and will run in the web browser (client).

Runtime kit

Open Source	Version	License	Web Site
Angular	1.5.8	MIT license	http://angularjs.org/
angular-ui-ace	0.2.3	MIT license	https://github.com/angular-ui/ui-ace
ace-builds	1.2.5	Copyright (c) 2010, Ajax.org B.V. All rights reserved.	https://github.com/ajaxorg/ace-builds
angular-bootstrap	1.1.2	MIT license	https://github.com/angular-ui/bootstrap
angular-cache	4.6.0	MIT license	https://github.com/jmdobry/angular-cache
angular-cookies	1.5.8	MIT license	https://github.com/Elzair/angular-module-cookies
angular-dynamic-locale	0.1.32	MIT license	https://github.com/lgalfaso/angular-dynamic-locale
Angular-leaflet-directive	0.10.0	MIT license	https://github.com/tombatossals/angular-leaflet-directive
angular-resource	1.5.8	MIT license	https://github.com/roylines/node-angular-resource
angular-route	1.5.8	MIT license	https://github.com/Elzair/angular-module-route
angular-sanitize	1.5.8	MIT license	https://github.com/Elzair/angular-module-sanitize
angular-schema-form	0.8.13	MIT license	https://github.com/Textalk/angular-schema-form
angular-translate	2.11.1	MIT license	https://github.com/angular-translate/angular-translate
angular-translate-loader-partial	2.11.1	MIT license	https://github.com/angular-translate/bower-angular-translate-loader-partial
angular-tree-control	0.2.28	MIT license	https://github.com/wix/angular-tree-control
angular-ui-grid	3.2.1	MIT license	https://github.com/angular-ui/ng-grid/blob/master/LICENSE.md
bootstrap	3.3.7	MIT license	http://getbootstrap.com/
checklist-model	0.10.0	MIT license	http://vitalets.github.io/checklist-model
es5-shim	4.5.9	MIT license	https://github.com/es-shims/es5-shim
font-awesome	4.6.3	CC-BY-3.0, MIT	https://github.com/FortAwesome/Font-Awesome
Jasny-bootstrap	3.1.3	Apache License 2.0	https://github.com/jasny/bootstrap

jquery	3.1.0	Copyright 2005, 2014 jQuery Foundation and other contributors, https://jquery.org/	
jquery-knob	1.2.13	MIT license	https://github.com/aterrien/jQuery-Knob
jquery-ui	1.12.1		https://jqueryui.com/
javascript-state-machine	2.3.5	Copyright (c) 2012, 2013, 2014, 2015, Jake Gordon and contributors	https://github.com/jakesgordon/javascript-state-machine
JointJS	0.9.7	Mozilla Public License Version 2.0	http://jointjs.com https://github.com/clientIO/joint
json3	3.3.2	MIT license	http://bestiejs.github.io/json3/
highcharts-ng	0.0.12	MIT license	https://github.com/pablojim/highcharts-ng
requireJS	2.3.2	MIT license / BSD	http://requirejs.org
Leaftlet	1.0.1	Copyright (c) 2010-2016, Vladimir Agafonkin Copyright (c) 2010-2011, CloudMade All rights reserved.	https://github.com/Leaflet/Leaflet
Leaftlet.markercluster	1.0.0	MIT license	https://github.com/Leaflet/Leaflet.markercluster
Leaflet-plugins	1.9.3	Copyright (c) 2011-2015, Pavel Shramov, Bruno Bergot	https://github.com/shramov/leaflet-plugins
lodash	3.10.1	Copyright 2012-2014 The Dojo Foundation	https://github.com/lodash/lodash
lodash-deep	1.6.0	MIT license	https://github.com/marklagendijk/lodash-deep
moment	2.15.1	Copyright (c) 2011-2016 Tim Wood, Iskren Chernev, Moment.js contributors	https://github.com/moment/moment
ng-idle	1.3.1	MIT license	https://github.com/HackedByChinese/ng-idle
ng-tags-input	3.1.1	MIT license	https://github.com/mbenford/ngTagsInput
require-css	0.1.8	MIT license	https://github.com/guybedford/require-css
requirejs-plugins	1.0.3	MIT license	https://github.com/millermedeiros/requirejs-plugins
proj4	2.3.15	MIT license	https://github.com/OSGeo/proj.4
text	2.0.15	Copyright jQuery Foundation and other contributors, https://jquery.org/	https://github.com/requirejs/text
socket.io-client	1.4.8	MIT license	https://github.com/socketio/socket.io-client
slickgrid	2.3.2	Copyright (c) 2009-2016 Michael Leibman,	https://github.com/mleibman/SlickGrid https://github.com/6pac/SlickGrid
Summernote	0.8.2	MIT license	http://summernote.org/ https://github.com/summernote/summernote

Figure 110: Open Sources integrated in UOC V2.3 (Client)

SDK kit

Open Source	Version	License	Web Site
Angular-mock	1.5.8	MIT license	https://github.com/angular/bower-angular-mocks
Chai	3.5.0	MIT license	https://github.com/chaijs/chai
Sinon	1.17.5	BSD license	https://github.com/sinonjs/sinon
Sinon-chai	2.8.0	WTFPL and BSD licenses	https://github.com/domenic/sinon-chai
Chai-as-promised	5.3.0	WTFPL license	https://github.com/domenic/chai-as-promised

Chapter 24
External References

You can find more details and information on the following links:

- Apache CouchDB: http://couchdb.apache.org/
- NodeJS: http://nodejs.org/
- Redis: http://redis.io/
- Socket.io: http://socket.io/
- AngularJS: https://angularjs.org/
- Bootstrap: http://getbootstrap.com
- Npm: https://www.npmjs.com/
- Grunt: http://gruntjs.com/
- Bower: http://bower.io/
- Yeoman: http://yeoman.io/
- Log4js: https://github.com/nomiddlename/log4js-node
- SublimeText : https://www.sublimetext.com/
- SublimeText Package Manager: https://packagecontrol.io
- JSONLint : http://jsonlint.com/
- Csurf : https://github.com/expressjs/csurf
- Helmet: https://github.com/helmetjs/helmet
- Helmet / CSP: https://github.com/helmetjs/csp
- Helmet / Hsts: https://github.com/helmetjs/hsts
- Helmet / Frameguard: https://github.com/helmetjs/frameguard
- Node-Sass: https://github.com/sass/node-sass
- Sass: http://sass-lang.com/
- Highcharts : http://www.highcharts.com/
- NSP: https://github.com/nodesecurity/nsp
- Node Security Advisories: https://nodesecurity.io/advisories
- JSON Schema: http://json-schema.org/