ngEO Task 4

Web Client

Detailed Design Document

document status sheet

| Version | Date | Pages | Changes |
| --- | --- | --- | --- |
| 0.1 | 23/10/2012 | 31 | First published version |
| 0.2 | 25/10/2012 | 34 | DDR version |
| 1.0 | 29/10/2012 | 34 | Updated company name, date and version in footer |
| 1.1 | 20/12/2012 | 34 | Updated data access requests design |
| 1.2 | 18/01/2013 | 35 | First CDR delivery |
| 1.3 | 28/01/2013 | 35 | Second CDR delivery |
| 1.4 | 05/03/2013 | 40 | Delivery after CDR rids.  [NGEOC-306](https://magelliumltd.atlassian.net/browse/NGEOC-306): Missing interface details. Sequence diagram added in 5.3 |
| 1.5 | 16/10/2013 |  | DDR V2 draft delivery |

TABLE OF CONTENTS

1 Introduction 7

2 Applicable and Reference Documents 8

2.1 Applicable Documents 8

2.2 Reference Documents 8

3 Terms, Definitions and Abbreviated Terms 9

3.1 Definitions 9

3.2 Acronyms 9

4 Software Design Overview 10

4.1 Software static architecture 10

4.2 Software dynamic architecture 11

4.3 Interfaces context 11

4.4 Long lifetime software 12

4.5 Memory and CPU budget 12

4.6 Design standards, conventions and procedures 13

4.6.1 Software architectural design method 13

4.6.2 Code documentation standards 13

5 Software design 14

5.1 General 14

5.2 Overall architecture 14

5.3 Software components design - General 15

5.3.1 Modular javascript: RequireJS 15

5.3.2 User Interface 16

5.3.2.1 Core 16

5.3.2.2 MenuBar 17

5.3.2.3 DataServicesArea 17

5.3.3 Map 17

5.3.4 Model/View 18

5.3.4.1 Backbone 18

5.3.4.2 Search 19

5.3.4.2.1 Modules 19

5.3.4.2.2 Interactions 20

5.3.4.3 SearchResults 22

5.3.4.3.1 Modules 22

5.3.4.3.2 Interactions 22

5.3.4.4 Account 23

5.3.4.4.1 Modules 23

5.3.4.4.2 Interactions 23

5.3.4.5 DataAccess 24

5.3.4.5.1 Modules 24

5.3.4.5.2 Interactions 25

5.3.4.6 Shopcart 25

5.3.5 Configuration 26

5.3.6 Simulator 26

5.4 Software components design - Aspects of each component 26

5.4.1 MenuBar 26

5.4.1.1 Type 26

5.4.1.2 Purpose 26

5.4.1.3 Function 26

5.4.1.4 Subordinates 26

5.4.1.5 Dependencies 26

5.4.1.6 Interfaces 27

5.4.1.7 Resources 27

5.4.1.8 Data 27

5.4.2 DataServicesArea 27

5.4.2.1 Type 27

5.4.2.2 Purpose 27

5.4.2.3 Function 27

5.4.2.4 Subordinates 27

5.4.2.5 Dependencies 27

5.4.2.6 Interfaces 27

5.4.2.7 Resources 27

5.4.2.8 Data 27

5.4.3 Map 27

5.4.3.1 Type 27

5.4.3.2 Purpose 28

5.4.3.3 Function 28

5.4.3.4 Subordinates 28

5.4.3.5 Dependencies 28

5.4.3.6 Interfaces 28

5.4.3.7 Resources 28

5.4.3.8 Data 28

5.4.4 Search 28

5.4.4.1 Type 28

5.4.4.2 Purpose 28

5.4.4.3 Function 28

5.4.4.4 Subordinates 28

5.4.4.5 Dependencies 29

5.4.4.6 Interfaces 29

5.4.4.7 Resources 29

5.4.4.8 Data 29

5.4.5 SearchResults 29

5.4.5.1 Type 29

5.4.5.2 Purpose 29

5.4.5.3 Function 29

5.4.5.4 Subordinates 29

5.4.5.5 Dependencies 29

5.4.5.6 Interfaces 29

5.4.5.7 Resources 29

5.4.5.8 Data 29

5.5 Internal interface design 30

5.5.1 DataServicesArea 30

5.5.2 MenuBar 30

5.5.3 Map 31

5.5.4 Search 32

6 Requirements to design components traceability 33

6.1 Requirements to Components 33

6.2 Components to Requirements 38

list of tables and figures

Table 2‑1 Applicable Documents 8

Table 2‑2 Reference Documents 8

Table 3‑1 Definitions 9

Table 3‑2 Acronyms 9

Figure 1 Software Design Overview 11

Figure 2 ngEO Web User Client Architecture 15

Figure 3 Map facade 18

Figure 4 Search component 20

Figure 5 DataSet population sequence diagram 21

Figure 6 Dataset selection sequence diagram 22

Figure 7 Search results sequence diagram 23

Figure 8 Download manager monitoring sequence diagram 24

Figure 9 Simple Data Access sequence diagram 25

# Introduction

This document has been created taking as a base the corresponding DRD (Document Requirements Definition) in [AD.1].

This document defines the architectural design for the ngEO Web Client subsystem.

The Web Client is the user interface of the ngEO system and is also the main visible part of the whole system. The main target of the design is to provide an easy to use interface and also highly responsive in order to receive positive feedback from the end user on the entire system. It is also important that the interface is based on modern design and easily configurable to ensure it will not be obsolete too early during the long operational life of ngEO (20 years).

The Web Client allows a user to:

* Search products in available Datasets :
  + Discover and select Datasets directly or through filters (mission/sensor/keywords)
  + Search products in a selected Dataset using various criteria (spatial, temporal, specific)
  + Advanced search : correlation and interferometry
* Explore search results :
  + List them in a tabular view
  + Display them in a GANTT chart
  + Display their footprint in a map
  + Display their browse images in a map
* Download products :
  + Direct download through native browser functionality
  + Retrieve products via ngEO Download Manager through a data access request
  + Configure optional processing options for products retrieved through download manager
  + Configure “Standing Orders” : schedule download of products through specific criteria
  + Monitor download manager and data access request
* Manage shop carts :
  + Store a list of products from a search results in a shopcart for later reuse
  + Share shopcart with other users
* Access to various user related features :
  + Access to account management page on the UM-SSO Idp
  + Access to the ticketing system
  + Access to information page (About Us)
* Manage customizable Web client configuration

# Applicable and Reference Documents

## Applicable Documents

The following documents, of the exact issue shown, form part of this document to the extent specified herein. Applicable documents are those referenced in the Contract or approved by the Approval Authority. They are referenced in this document in the form [AD.X]:

Table 2‑1 Applicable Documents

| Ref. | Title | Code | Version | Date |
| --- | --- | --- | --- | --- |
|  | Space Engineering - Software | ECSS-E-ST-40 | C | 6 March 2009 |
|  | INTERNAL INTERFACE CONTROL DOCUMENT WEB CLIENT - WEB SERVER | ngEO-14-ICD-ELC-075 | 1.3 | 16 August 2012 |
|  | EXTERNAL INTERFACE CONTROL DOCUMENT FOR NGEO WEB SERVER |  | 1.4 | 03 July 2012 |
|  | EXTERNAL INTERFACE CONTROL DOCUMENT FOR NGEO BROWSE SERVER | ngEO-14-ICD-SPM-089 | 1.2 | 15 January 2012 |
|  | SUBSYSTEM REQUIREMENT DOCUMENT FOR NGEO WEB USER CLIENT | ngEO-14-SRD-ELC-005 | 1.3 | 16 August 2012 |
|  | NGEO SYSTEM REQUIREMENT DOCUMENT | ngEO-13-SRD-MFR-004 | 1.6 | 21 August 2012 |
|  | SYSTEM ARCHITECTURAL DESIGN DOCUMENT |  | 1.31 | 6 July 2012 |

## Reference Documents

The following documents, although not part of this document, amplify or clarify its contents. Reference documents are those not applicable and referenced within this document. They are referenced in this document in the form [RD.X]:

Table 2‑2 Reference Documents

| Ref. | Title | Code | Version | Date |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

The following list of web references is for general information only.

[WR1] RequireJS, <http://requirejs.org/>

[WR2] jQuery, <http://jquery.org/>

[WR3] jQuery Mobile, <http://jquerymobile.com/>

[WR4] LESS, <http://lesscss.org/>

[WR5] Node.js, <http://nodejs.org/>

[WR6] OpenLayers, <http://openlayers.org/>

[WR7] GlobWeb, <https://github.com/TPZF/GlobWeb>

# Terms, Definitions and Abbreviated Terms

## Definitions

Concepts and terms used in this document and needing a definition are included in the following table:

Table 3‑1 Definitions

| Concept / Term | Definition |
| --- | --- |
|  |  |

## Acronyms

Acronyms used in this document and needing a definition are included in the following table:

Table 3‑2 Acronyms

| Acronym | Definition |
| --- | --- |
| AMD | Asynchronous Module Definition |
| CSS | Cascading Style Sheet |
| DOM | Document Object Model |
| JSON | Javascript Object Notation |
| GUI | Graphical User Interface |
| HTML | Hyper Text Markup Langague |
| LESS | Dynamic Stylesheet |
| MVC | Model View Controller |
| OGC | Open Geospatial Consortium |
| REST | Representational State Transfer |
| WMTS | Web Mapping Tile Service |

# Software Design Overview

## Software static architecture

The Web Client software architecture is made of several components; each of them contains JavaScript, HTML and CSS code. These components are integrated into a single web application delivered by an Apache httpd server.

The Web Client components are:

* User Interface: manage the main user interface elements: Home Page, MenuBar, ToolBar, ContextHelp and Widget.
* Map: manage the map view, the different actions in toolbar (zoom, draw..) and interaction with third-party geoprovider
* Model/View: regroup components that manage the business logic of the application and interactions with WebServer:
  + Account: Manage the account information of the current logged user (Name, Category, Download Managers, Data Access Requests, Shopcarts)
  + Search: Manage product search in a dataset, dataset selection, search criteria
  + SearchResult: Manage display of search results in table view and Gantt chart
  + Shopcart: Manage the shopcart
  + Data Access: Manage DAR, Download Manager, Standing Order and download options
* Configuration: store the Web Client configuration through a map of Javascript objects, one for each module that needs some configuration options.

The main page layout consists of the following elements:

* A menu bar to display different navigation links
* A tool bar to access widgets (Search, Layers, Shopcarts) and user actions on the map (zoom, draw…)
* A main central area covering the whole available space and filled by the map

The user interface is really map-centric as described in [AD.5].



Figure 1 Software Design Overview

## Software dynamic architecture

The Web User Client interacts mainly with the Web Server. All communications between Web User Client and Web Server are based exclusively on REST principles and using JSON as data representation. It gives a simple, yet powerful interface between the client and server.

Apart from the interaction with the Web Server, the Web User Client will also integrate geospatial data coming from different sources based on standard OGC protocol (WMS, WFS, WMTS, GeoRSS..) or specific services (Gazetteer). A special provider of geospatial data is the ngEO Browse Server to display browse images of products. The client communicates with Browse Server using WMTS (Web Map Tile Service) to reduce latency.

## Interfaces context

The Web Client exchanges with two other ngEO subsystems: Web Server and Browse Server.

The interfaces exchanged between the ngEO Web Client and the other subsystems are inputs for task 4 and are specified in the following ICDs:

* IICD-WC-WS ([AD.2]): defining all interfaces between the Web Server and Web Client.
* EICD-WS ([AD.3]): defining all external interfaces of Web Server, especially OpenSearch based service for product search used by the Web Client.
* EICD-BS ([AD.4]): defining the interfaces for WMTS access to browse layer.

Here we provide a short but complete summary of theses interfaces for each internal component of the Web Client subsystem:

* Configuration:
  + IF-ngEO-WebClientConfigurationData: retrieves the configuration data, contains configuration for each component.
* Search:
  + IF-ngEO-DatasetSearchInfo : retrieves the information of a dataset given by its Id
  + IF-ngEO-DatasetPopulationMatrix: retrieves the dataset population to initialize dataset chooser
  + IF-ngEO-DataSetAuthorization: retrieves all relevant authorization information (search access, browse,download,hosted processing) for datasets
* SearchResults:
  + IF-ngEO-CatalogueSearch: issues a product search against selected dataset following OpenSearch specification
* Shopcart:
  + IF-ngEO-ShopCartManagement: regroups seven interfaces to manage the shop carts and the items contained in them. It basically provides CRUD operations on shop carts and items
  + IF-ngEO-ShopCartContent: retrieves the content of a shopcart
* Account:
  + IF-ngEO-UserShopCartsConfigData: retrieves the list of shopcarts belonging to requested user
  + IF-ngEO-UserConfigurationData: retrieves the user specific data
  + IF-ngEO-UserDlManagersConfigData: retrieves the download managers associated with the user
  + IF-ngEO-UserDataAccessReqConfigData: retrieves the Data Access Request belonging to a user
  + IF-ngEO-RestrictedAccessRequest: requests an access to restricted dataset for a user
  + IF-ngEO-TicketCreation: interface provided by the Ticketing System of CDS and used by ngEO Web Server for asking the creation of a ticket. The Web Server needs to provide a similar interface to the web client in order to submit inquiry.
* Data Access:
  + IF-ngEO-DataAccessRequest: triggers a data access request
  + IF-ngEO-DataAccessRequestStatus: used to know the status a specific data access request
  + IF-ngEO-HostedProcessingList: retrieves the list of hosted processes supported for each product in the supplied list of products – this list may be different for each product in the list.
  + IF-ngEO-DownloadManagerChange: This flow is used to order a Download Manager to stop
* Map:
  + IF-ngEO-WMTSBrowseAccess: gives WMTS service used by WebClient to display browse image

## Long lifetime software

The Web Client is built on the most recent HTML standard (HTML5) supported by many different companies and institutions (W3C). Compatibility is assured for the excepted lifetime of the ngEO system.

## Memory and CPU budget

The Web Client is executed on the user computer. The following minimal requirements need to be met in terms of memory and CPU budget on the user computer:

* 1 GB RAM
* 1 CPU
* No disk space requirement

## Design standards, conventions and procedures

This section summarizes the software methods adopted for the architectural and the detailed design.

### Software architectural design method

UML is used as the modelling language: UML is unambiguous and therefore very effective for communication within the team and outside of the team. Version 2.0 of UML is used in this project.

### Code documentation standards

JSDoc (Documentation generator for Javascript) is used for code documentation. See <http://code.google.com/p/jsdoc-toolkit/w/list> for the list of tags.

# Software design

## General

The Web Client software is based on HTML/CSS/Javascript technologies. Javascript is the central technology, HTML and CSS being resources manipulated by Javascript code. Web application are now becoming more and more complex and design methodology being used for languages as Java are now used to develop an easy to maintain web application.

Javascript is a very flexible oriented object language but it does not have a “class” concept. Nevertheless, Javascript can simulate the class concept using well known methodology (function constructor to summarize). In the rest of the document, we are going to speak about class even if technically it is a little bit different. The paragraph 5.3.1 describes more precisely how to build modular and easy to maintain web applications in Javascript.

## Overall architecture

The following picture reports the first level decomposition of ngEO Web User Client components.

The components are already described in paragraph 4.1.



Figure 2 ngEO Web User Client Architecture

## Software components design - General

### Modular javascript: RequireJS

An application is said to be modular if it is composed of a set of highly decoupled, distinct pieces of functionality stored in modules. Loose coupling facilitates easier maintainability of applications by removing dependencies where possible. With a modular system, it is quite easy to see how changes to one part of a system may affect another and is fundamental when dealing with a medium to large code base.

Unlike some more traditional programming languages however, Javascript doesn't provide developers with the means to import such modules of code in a clean, organized manner. The traditional approach is to integrate Javascript files using script tag in the main HTML page. There is no clean way to handle dependency management.

Hopefully, Javascript developer’s community has built new and standardised solutions to this problem. Among these solutions, RequireJS ([WR.1]) is used for ngEO Web Client. RequireJS is an implementation of the Asynchronous Module Definition (AMD) specification. This specification is supported by various projects: Dojo (1.7), MooTools (2.0), Firebug (1.8) and jQuery (1.7).

The AMD specification is powerful but still quite simple. The two key concepts are the idea of a *define* method for facilitating module definition and handling dependency loading. The main entry point of the applications uses *require* for all main dependencies.

For ngEO Web Client, a module can have the following behaviour:

* Return a plain object, with different methods, identical to a singleton in Java;
* Return a function constructor, identical to a class in Java;
* Define a jQuery plugin, registered into the jQuery global object; return nothing in that case.

### User Interface

#### Core

The component consists of a home page that defines the main layout of the user interface:

* A menu bar to display different pages ;
* A main central area covering the whole available space and filled by the map.

When user access to “Data Services Area”, a toolbar is displayed above the map to give access to different services (Search, Layers, Shopcarts) and user actions on the map (zoom, background, switch between 2D and 3D).

The main layout is based on the following libraries:

* jQuery for DOM manipulation and effects [WR.2]
* jQuery Mobile for widgets (listview, button, form elements) [WR.3]
* Custom stylesheets based on LESS technology [WR.4]

The three main libraries are described in subsequent paragraphs.

jQuery is a Javascript library designed to simplify web client development. jQuery is recognized as one of the most popular libraries today (see http://w3techs.com/technologies/overview/javascript\_library/all) due mainly to its ease of use and clear design.

The library is not a complete framework with a full set of predefined widgets but instead focus on simplifying the use of web client core technologies and hiding the differences between browsers.

The main features of jQuery are:

* HTML document traversal, selection and manipulation
* Events handling
* Effects and animations
* AJAX interactions
* Extensibility through plugins

jQuery Mobile is a touch-optimized JavaScript library build on the jQuery and jQuery UI foundation. The focus of the library is to create a compatible mobile user interface across the heterogeneous market of tablet and smartphone. The user interfaces is really modern and touch-ready and use as much as possible the latest HTML5 and CSS3 technologies. The UI paradigm of the library is really targeted to mobile device. Some adaptations are needed to integrate jQuery Mobile into the Web Client UI but it provides a modern look and feel.

LESS extends CSS with dynamic behaviour such as variables, mix-ins, operations and functions. During development, LESS is converted client-side using the LESS Javascript library that automatically converts LESS into normal CSS that is interpreted by the browser. For production, LESS files are precompiled into CSS files during a build phase.

#### MenuBar

The MenuBar component is a custom ngEO jQuery UI plugin, based on existing design from TPZ France. A MenuBar is applied on an existing DOM element containing link element (<a>) enhanced with various data attributes to describe how to display the page and how to load them.

The MenuBar is responsible to:

* Display page using AJAX if needed
* Style dynamic page to appears transparently above the main layout (menu bar and map)
* Dynamically load Javascript module using indication on a link element

The MenuBar is an important component in the software architecture because it allows easily removing or adding of components thanks to dynamic loading of Javascript modules.

#### DataServicesArea

The DataServicesArea module is made of 2 sub components:

* ToolBar: a simple component to give access to the different services of the WebClient. The ToolBar is DOM, with some specific styling applied by the DataServicesArea module.
* Widget: this component is a custom ngEO jQuery UI plugin to manage widgets in the WebClient. Widgets can be linked to buttons on the toolbar (same as iPad menu widgets), or used directly as a modal dialog.
* PanelMananager : this component is a custom Backbone view used to manage panels in the WebClient, based on existing design from TPZ France. Panels can be placed in different areas of a main container with a central area. In ngEO, the central area is the map.

The Widget module provides a common interface for dealing with mostly popup widgets, and provides the following functionality:

* Button management in a specific footer area
* Activation from a button in the main toolbar (optional)
* Title management (optional)
* Close button (optional)
* Modal-like behaviour (no access to other GUI elements when opened)
* Show/hide with animation

The PanelManager module provides mostly the same functionality, and shares the same styling to have a coherent appearance in the application:

* Button management in a specific footer area
* Activation from a button in the main toolbar (optional)
* Show/hide

These components have been designed in order to be able to add easily a new service in the data services area.

### Map

The main map module is based on existing design from Telespazio France for its own development of complex web applications. The component is adapted to fit ngEO needs. This component is a facade to underlying implementations based on existing solutions: OpenLayers ([WR5]) and GlobWeb ([WR6]). OpenLayers is for 2D visualization and GlobWeb for 3D visualization based on WebGL. Underlying implementations are not known from other components point of view.

The component is a Javascript module and uses the GeoJSON specification for feature management.

This component has the following features:

* Common viewpoint management between 2D and 3D implementations
  + Zoom to a point, to an extent, to a feature
  + Save/restore viewpoint when switching between 2D and 3D
* Base layers configuration specific to each implementation through JSON options
* Common interface to add/remove/select vector programmatically in the map
  + Vector defined in the standard GeoJSON format
  + Selection and bounding box computation used by common code between implementation
  + “Pile” selection supported
* Communication done through Backbone.js events
  + Selection event
  + Viewpoint changed event

The component shares some common code to simplify the integration of implementation libraries. To be integrated, a library must support at least the following low-level features:

* Modification of visualized area (viewport) through geographic coordinates
* Access to the current viewport extent
* Conversion of element coordinates (x,y) to geographic (lon,lat)
* Creation and display of vector data : point, polyline, polygon



Figure 3 Map facade

The component also contains the GUI of map:

* Layer management widget
* Background widget
* Different map actions in the toolbar
  + ZoomInAction
  + ZoomOutAction
  + DrawAction
  + Switch2D3DAction
  + StartAction

### Model/View

#### Backbone

Backbone.js is a lightweight Open Source library that provides a clean way to structure a web application. The structure is loosely based on the classical MVC (Model/View/Controller) design patterns.

Backbone consists of five major components:

* Events: This module can add to any Javascripts objects the ability to bind and trigger custom named events with various parameters.
* Model: This class enhances the classical key/value objects in Javascript with events to detect modifications, and utilities methods for storing and retrieving data from the server, initializing default values, etc...
* Collection: This class manages an ordered set of models, with custom events, and basic saving/loading from a server based on RESTful principles.
* Router: This class provides methods for routing client-side pages, and connecting them to actions and events.
* View: This class provides the glue between the GUI (HTML/CSS) and the model. It is more convention that code, Backbone does not impose anything when developing a View. The class naming is a little bit confusing because it looks more as a Controller in traditional MVC.

Backbone.js is very well suited to ngEO Web Client architecture. The events module solves the communication requirements between widgets. The Router module also provides a cross-browser compatible mechanism to access a shopcart by its URL. Backbone will automatically call a function when a specific URL pattern (or route) is recognized. For example for shopcart, the following route might be used: “/shopcart/:id”. When the URL http://ngeo.org/#/shopcart/1026 is accessed, the registered function will be called with the id equals at 1026 in parameter. This function will send a request to the Web Server to retrieve the shopcart content and then display it in a widget.

Model in ngEO (DataSet, Product, Shopcart, DownloadManagers, DataAccessRequest, etc...) will use the Backbone lightweight Model class.

This library is used in various commercial website, in particular LinkedIn Mobile or WallMart Mobile.

Backbone also integrates the underscore.js library. It is a small utility library that provides more functional programming support to Javascript, and a light-weight client-side HTML templating engine, similar to JSP on server-side. The template engine is useful to generate complicated bits of HTML from JSON data sources.

#### Search

##### Modules

The component is made of the following javascript modules for the model part:

* DataSet: a Javascript class that contains all datasets informations needed by the WebClient.
* DataSetSearch: It is a Javascript class that contains all the search parameters needed to search products in a DataSet. The class is also responsible to generate the OpenSearchURL from the search parameters. The class also validates if the properties are suitable for a search. The search parameters include also the selected download options if the user make the choice to include them in the search.
* DataSetPopulation: an object map retrieved form the server to quickly filter out the dataset when the user selects mission/sensor/keywords. The object has various methods to facilitate the dataset chooser view and provides autocomplete support.
* DatasetAuthorizations: an object to manage user authorization on datasets. Use by different views to show and inform user about its authorization on the datasets.

The component is made of the following javascript modules for the view part:

* TimeExtentView: this view controls the user interface to set the time extent. The user interface is based on several jQuery plugins: DateBox for calendar input, jQRangeDataSlider for time slider, a list of key-date widget retrieved from the server.
* SpatialExtentView: this view controls to set spatial extent. The user interface contains several elements : numeric input to enter bounding box coordinates, combo box to select a feature in the map (either drawn by the user or loaded from a file) and flag to use the current map extent
* AdvancedSearchView: this view controls all advanced criteria for the selected dataset if any. Each search criterion, according to its type and the number of values to select, is displays by a suitable widget; such as sliders for range types, radio boxes for single-valued criteria, a group of checkboxes for multiple-valued criteria.
* DownloadOptionsView: This view displays the download options of the selected dataset if any.
* DataSetSelectionView: this view controls the dataset selection

SearchCriteriaView: this view handles all the views to fill in for a dataset search: time, area, advanced criteria and download options views. The views are displayed as tabs.



Figure 4 Search component

##### Interactions

The following sequence diagram shows what is happening when the user clicks on the “Dataset” button in the data services area toolbar:

* DatasetSelectionView is shown
* To build the UI, DatasetSelectionView requests DatasetPopulation to fetch its data from the server
* DatasetPopulation launch an asynchronous request to the Web Server (IF-ngEO-DatasetPopulationMatrix)
* When the request is received, DatasetPopulation launch a “success” event
* DatasetSelectionView receives the event and build its UI



Figure 5 DataSet population sequence diagram

The following sequence diagram shows what is happening when the user selects a dataset in the DatasetSelectionView:

* DatasetSelectionView modify Dataset used by DatasetSearch model
* To update its parameters (start/stop date, advanced criteria, download options), DatasetSearch model ask Dataset model to fetch the information for this dataset
* Dataset model launch an asynchronous request to Web Server (IF-ngEO-DatasetSearchInfo)
* When the request is received, the Dataset launch a “success” event
* DatasetSearch updates its attributes according to Dataset attributes and launch a change event for each attribute modification
* DatasetSelectionView receives the change event and updates its UI



Figure 6 Dataset selection sequence diagram

#### SearchResults

##### Modules

The component is made of the following Javascript modules for the model part:

* SearchResults: a Javascript class that manages search results and request to the WebServer using OpenSearchURL. The class is also responsible for background loading and paging.It does also manage the selected products and does contain needed methods on product urls’ checks and updates.

The component is made of the following Javascript modules for the view part:

* SearchResultsTable: manage the display of products in a table using jQuery DataTables plugin
* SearchResultsGantt: manage the display of products in a Gantt Chart using jQuery Gantt
* DownloadOptionsWidget: widget activated through the “Download Options” button of the SearchResultsTable view. It spawns up the DownloadOptionsWidgetView.
* DownloadOptionsWidgetView : is the view displayed inside the DownloadOptionsWidget and which contains all the download options available for the current dataset to which beling the checked products. The user can update the download options and the selected values are taken into account in each of the product urls.

##### Interactions

The following sequence diagram shows what is happening when the user click on the search button:

* The search button calls the launch method on SearchResults
* SearchResults send an event “reset:features” to tell any views (Map, SearchResultsTableView) to remove previous results
* SearchResults retrieves the current OpenSearch URL for the search on the DatasetSearch model
* SearchResults send an asynchronous request to Web Server to retrieve results for the first page
* When results are receveid, SearchResults send an event to tell the views that features have been added
* When SearchResultsTableView receive the event, the table is filled with the new features



Figure 7 Search results sequence diagram

#### Account

##### Modules

The component is made of:

* Javascript modules : Account models and views
* HTML page for the main Account user interface with a CSS

The Account model is composed of several classes: each class extends the Backbone.Model and contains all properties of the related ngeo Account object. These models are the following:

* DataAccessRequestStatuses is a Backbone model responsible for retrieving all the status information for the already submitted dataAccessRequests.
* DownloadManagers is a Backbone model responsible for retrieving all the download manager information for the already registered download managers.
* Shopcarts is a Backbone model responsible for retrieving all the shopcarts information for the shopcarts authorized to the logged in user.
* UserPrefs is a module responsible for storing and getting preferences to/from the local storage of the browser if it is supported.

Account.js is the main class handling all the account views as tabs.

Different views manage each aspect of the Account:

* DownloadManagersMonitoringView : presents a list of download managers available for a user
* DataAccessRequestMonitoringView : presents a monitoring of the DataAccesssRequests launched by the user
* InquiriesView : displays a form with inquiries possible types and a text area to write the message. A button submit, make the request submitted to the server.
* UserPrefsView : Lists the last selected dataset and background layer.
* UpgradeView : presents a form to upgrade authorization available to a user

##### Interactions

The following sequence diagram shows what is happening when the user click on the Download Manager Tab in the “My Account” page:

* DownloadManagersMonitoringView is shown, first it call fetch on DownloadManagers
* DownloadManagers send an asynchronous request to Web Server to retrieve the list of download managers for the current user
* DownloadManagersMonitoringView builds it content when it receives the event loaded() from DownloadManagers



Figure 8 Download manager monitoring sequence diagram

#### DataAccess

##### Modules

The component is made of the following Javascript modules for the model part:

* DataAccessRequest : a Javascript singleton mother class for SimpleDataAccessRequest and StandingOrderDataAccessRequest. It handles the common aspects of data access submission:
  + the assignment of a data access request to a download manager
  + the submission of a request to the server and the responses analysis
  + the testing of bulk order status
* SimpleDataAccessRequest: a Javascript singleton that manages simple data access request. The properties follow the JSON representation expressed in IICD-WC-CS
* EnhancedDataAccessRequest: a Javascript class that manages enhanced data access request. The properties followed the JSON representation expressed in IICD-WC-CS
* StandingOrderDataAccessRequest: a Javascript singleton that manages standing order data access request. The properties followthe JSON representation expressed in IICD-WC-CS
* Download Managers: a Backbone singleton model that manages all the information on download managers this same model is used for the account module.

The component is made of the following Javascript modules for the view part:

* HostedProcessChooser: a view to choose the hosted process
* HostedProcessParameterView: a view to configure an hosted process
* DownloadManagerListView: a view to choose a download manager for a given request
* DownloadManagerWidget: a popup widget displaying the DownloadManagerListView when simple data access requests are triggered.
* StandingOrderView : a view to choose either a data-driven or a time-driven standing order type and then enter the required parameters for the chosen type.

StandingOrderWidget: a popup widget displaying the StandingOrderView, and when the request is created it displays the DownloadManagerListView.

##### Interactions

The following sequence diagram shows what is happening when the user click on “Retreive Product” button:

* DataAccessRequestWidget is shown
* DataAccessRequestWidget initialize the SimpleDataAccessRequest with the current selection
* The user clicks on the Validate button in DataAccessRequestWidget, it call validate on SimpleDataAccessRequest
* SimpleDataAccessRequest send an asynchronous request to the Web Server to validate the data access request
* When the response is returned, DataAccessRequestWidget is notified through a SimpleDataAccessRequest event, it shows a message to user
* The same sequence is redone for confirmation.



Figure 9 Simple Data Access sequence diagram

#### Shopcart

##### Modules

The component is made of the following Javascript modules for the model part:

* Shopcart: a class to manage items on a Shopcart, provides the interface to load the shopcart, add/delete items through server requests.
* ShopcartCollection : manage the shop carts owned by the user, and also manage the current shopcart used in the data services area.

The component is made of the following Javascript modules for the model part:

* ShopcartManager : manage the display of the user shopcart list in “My Account” page
* ShopcartItemView: manage the display of shopcart, i.e. the shopcart items in a view, listen to Shopcart events to keep up to date the table view.

##### Interactions



Figure 10 Shopcart Sequence Diagram

### Configuration

The component is made of one Javascript module.

This module contains a Javascript object that is used as dictionary for all configuration option of modules in the application. The module is easily accessible for all modules of the applications.

Configuration options are just a collection of key/value pairs, i.e. a Javascript object.

For map module for example, the map options will contains the layers to display, style information to display footprints, etc...

### Simulator

A simulator is implemented in order to be able to develop and test the Web Client without the Web Server. The simulator is based on node.js ([WR.5]), a JavaScript solution on the server. This is to be coherent for all the sub-system: everything is written in JavaScript.

The simulator is kept very basic. Generally it always return the same JSON data when an interface is called, based on various input data files. The Express framework is used to route URLs to specific callbacks.

## Software components design - Aspects of each component

### MenuBar

#### Type

The component contains a Javascript module, following AMD definition and a Less file.

#### Purpose

See 6.2 for requirements.

#### Function

The components give access to different pages from the home page. The pages are loaded asynchronously when the user selects an item from the menu bar. The pages are cached to avoid reloaded them once requested by the user.

#### Subordinates

No subordinates

#### Dependencies

No dependencies.

#### Interfaces

Need a DOM element to insert it in the page.

Provide access to different components of the application.

#### Resources

None

#### Data

None

### DataServicesArea

#### Type

The component contains Javascript modules, following AMD definition and a Less files.

#### Purpose

See 6.2 for requirements.

#### Function

The components give access to the user at different services. It displays a list of buttons that will open custom widgets for each service.

#### Subordinates

No subordinates

#### Dependencies

It is called once the main layout is loaded.

#### Interfaces

Provide entry point to add easily some new services through the toolbar element.

#### Resources

Image for each services in the toolbar.

#### Data

None.

### Map

#### Type

The component contains multiple JavaScript modules, following AMD definition and a Less file.

#### Purpose

See 6.2 for requirements.

#### Function

The component provides 2D and 3D map visualization for the Web Client.

#### Subordinates

No subordinates

#### Dependencies

The component depends on ToolBar to provide the different actions a user can do on the map: zoom, layers management, background map selection....

#### Interfaces

Need a DOM element to insert it in the page.

Provides interface to:

* Get the current location on the map
* Display product footprints
* Display browse layer
* Modify current location

#### Resources

None

#### Data

None

### Search

#### Type

The component contains mutiple Javascript module, following AMD definition and a Less file.

#### Purpose

See 6.2 for requirements.

#### Function

The component provides dataset selection, search products in a dataset, and manage search criteria for a dataset.

#### Subordinates

No subordinates

#### Dependencies

The component depends on ToolBar to give access to its widgets.

#### Interfaces

Provides interface to:

* Listen when a search is requested by the user

#### Resources

None

#### Data

None

### SearchResults

#### Type

The component contains mutiple Javascript module, following AMD definition and a Less file.

#### Purpose

See 6.2 for requirements.

#### Function

The component provides visualization of search results through table, gantt chart and in map.

#### Subordinates

No subordinates

#### Dependencies

The component depends on ToolBar to give access to its widgets.

The component depends on Map to display result footprints and browse layer.

The component depends on Search to listen for search results.

#### Interfaces

None

#### Resources

None

#### Data

None

## Internal interface design

### DataServicesArea

Interface ELEMENT :

* Type : HTML markup
* Description : the basic markup to initialize the ToolBar
* Usage:

<div id="toolbar">

<img id="search" src="images/search.png" name="Search" />

<img id="shopcart" src="images/shopcart.png" name="Shopcart" />

<img id="layers" src="images/layers.png" name="Layers" />

<div class="tb-separator"></div>

<img id="home" src="images/home.png" name="Start View" />

<img id="prev" src="images/left.png" name="Previous" />

<img id="next" src="images/right.png" name="Next" />

<img id="zoomOut" src="images/zoomOut.png" name="Zoom Out" />

<img id="zoomIn" src="images/zoomIn.png" name="Zoom In" />

<img id="background" src="images/background.png" name="Background" />

<img id="switch" src="images/switch.png" name="2D/3D" />

</div>

Interface addAction :

* Type : jQuery UI method
* Description : Add an action in the toolbar.
* Usage : widget.toolbar(“addAction”, {…});

### MenuBar

Interface ELEMENT :

* Type : HTML markup
* Description : Gives the basic markup to build MenuBar
* Usage:

<nav>

<a href="#about" data-page="pages/about.html">About</a>

<a href="#data-services-area" data-nowrap="true" data-module="data-services-area" class="active">Data Services Area</a>

<a href="#account" data-page="pages/account.html">My account</a>

<a href="#support" data-page="pages/support.html">Support</a>

</nav>

Interface LOAD\_MODULE :

* Type : data attribute on a DOM element
* Description : Load dynamically a module to implement page behavior.
* Usage : data-module=”module”

Interface LOAD\_PAGE :

* Type : data attribute on a DOM element
* Description : Load dynamically a static page.
* Usage : data-page=”path/to/page.html”

### Map

Interface getViewportExtent :

* Type : JavaScript Method
* Description : get the current map extent
* Parameters : None
* Return : an array of 4 floats [south,west,north,east]

Interface updateViewportSize:

* Type : JavaScript Method
* Description : must be called when window is resized by the main user interface
* Parameters : None

Interface addFootprint:

* Type : JavaScript Method
* Description : add a GeoJSON feature to be displayed into the map
* Parameters : A GeoJSON feature or feature collection

Interface removeFootprint:

* Type : JavaScript Method
* Description : remove a displayed GeoJSON feature from the map
* Parameters : A GeoJSON feature or feature collection

Interface addBrowseLayer:

* Type : JavaScript Method
* Description : add a browse layer in the the map
* Parameters : A browse layer description

Interface removeBrowseLayer:

* Type : JavaScript Method
* Description : remove a browse layer from the map
* Parameters : A browse layer description

Interface zoomTo:

* Type : JavaScript Method
* Description : zoom to the given feature
* Parameters : A GeoJSON feature or feature collection

Interface startNavigation:

* Type: Backbone Event
* Description : trigger when the user starts navigating in the map
* Context : None

Interface endNavigation:

* Type: Backbone Event
* Description : trigger when the user ends navigating in the map
* Context : None

### Search

Interface searchRequested

* Type: Backbone Event
* Description : triggers when the user requests a search
* Context : The view that request the search

# Requirements to design components traceability

## Requirements to Components

|  |  |  |  |
| --- | --- | --- | --- |
| **SSRD-WC** | **Requirement Title** | **A** | **Component** |
| ngEO-SUB-001-WEBC-DES | Web user interface | V1 | User Interface |
| ngEO-SUB-002-WEBC-DES | Web user interface | V1 | User Interface |
| ngEO-SUB-004-WEBC-DES | Web Client Layout | V1 | User Interface |
| ngEO-SUB-005-WEBC-DES | Widget Visibility | V1 | User Interface |
| ngEO-SUB-008-WEBC-DES | Widget reset | V1 | User Interface |
| ngEO-SUB-009-WEBC-DES | Widget communication | V1 | Backbone.js |
| ngEO-SUB-013-WEBC-DES | Shared Web interface context | V1 | User Interface/ Backbone.js |
| ngEO-SUB-014-WEBC-DES | Catalogue search | V1 | Search |
| ngEO-SUB-015-WEBC-DES | Web Client Orientation Page | V1 | User Interface |
| ngEO-SUB-016-WEBC-DES | Technology | V1 | All |
| ngEO-SUB-021-WEBC-DES | HTML5 Caching | V1 | Deployment |
| ngEO-SUB-022-WEBC-DES | Map | V1 | Map |
| ngEO-SUB-023-WEBC-DES | Map controls | V1 | Map |
| ngEO-SUB-024-WEBC-DES | Web client implementations | V1 | User Interface |
| ngEO-SUB-025-WEBC-DES | Multi-devices' considerations | V1 | User Interface |
| ngEO-SUB-028-WEBC-DES | Product meta-data | V1 | SearchResult |
| ngEO-SUB-030-WEBC-DES | Product footprints | V1 | Map |
| ngEO-SUB-031-WEBC-DES | Product footprints Display | V1 | Map |
| ngEO-SUB-033-WEBC-DES | Browse display | V1 | Map |
| ngEO-SUB-034-WEBC-DES | Footprint and browse selection | V1 | Map |
| ngEO-SUB-035-WEBC-DES | Overlay layers | V1 | Map |
| ngEO-SUB-036-WEBC-DES | Background maps | V1 | Map |
| ngEO-SUB-037-WEBC-DES | Gazetteers | V1 | Map |
| ngEO-SUB-038-WEBC-DES | Direct Download | V1 | DataAccess |
| ngEO-SUB-039-WEBC-DES | Shopcarts | V2 | Shopcart |
| ngEO-SUB-040-WEBC-DES | Shopcarts URL | V2 | Shopcart |
| ngEO-SUB-041-WEBC-DES | Shared shopcarts | V2 | Shopcart |
| ngEO-SUB-042-WEBC-DES | Shared shopcarts Loading | V2 | Shopcart |
| ngEO-SUB-043-WEBC-DES | Groups of search results | V1 | SearchResult |
| ngEO-SUB-045-WEBC-DES | Groups of search results for correlation | V1 | SearchResult |
| ngEO-SUB-046-WEBC-DES | Groups of search results for interferometry | V1 | SearchResult |
| ngEO-SUB-048-WEBC-DES | Opensearch URL | V1 | Search |
| ngEO-SUB-049-WEBC-DES | Opensearch URL Submitting | V1 | Search |
| ngEO-SUB-050-WEBC-DES | Standing Order initialisation | V2 | Search |
| ngEO-SUB-055-WEBC-DES | Authentication | V2 | Apache Httpd |
| ngEO-SUB-058-WEBC-DES | Initialisations | V1 | Search |
| ngEO-SUB-060-WEBC-DES | Dataset Definition refresh | V1 | Search |
| ngEO-SUB-061-WEBC-DES | Geographical Extent of the catalogue search | V1 | Map |
| ngEO-SUB-062-WEBC-DES | File exchange for Shopcart | V2 | Shopcart |
| ngEO-SUB-063-WEBC-DES | File exchange | V1 | All |
| ngEO-SUB-101-WEBC-FUN | Web Client homepage | V1 | User Interface |
| ngEO-SUB-102-WEBC-FUN | Data access services | V1 | User Interface |
| ngEO-SUB-103-WEBC-FUN | Data access services area | V1 | User Interface |
| ngEO-SUB-104-WEBC-FUN | Map mode | V1 | Map |
| ngEO-SUB-105-WEBC-FUN | Default map mode | V1 | Map |
| ngEO-SUB-106-WEBC-FUN | Map's enrichment | V1 | Map |
| ngEO-SUB-107-WEBC-FUN | selection of item on map | V1 | Map |
| ngEO-SUB-108-WEBC-FUN | zone of interest definition | V1 | Map |
| ngEO-SUB-109-WEBC-FUN | Map's navigation and zooming | V1 | Map |
| ngEO-SUB-110-WEBC-FUN | Map's layers management | V1 | Map |
| ngEO-SUB-111-WEBC-FUN | Dataset Chooser | V1 | Search |
| ngEO-SUB-112-WEBC-FUN | Dataset Chooser | V1 | Search |
| ngEO-SUB-113-WEBC-FUN | Dataset selection | V1 | Search |
| ngEO-SUB-114-WEBC-FUN | Dataset selection service | V1 | Search |
| ngEO-SUB-115-WEBC-FUN | Default layers for the Map Widget | V1 | Map |
| ngEO-SUB-116-WEBC-FUN | Dataset selection dynamic | V1 | Search |
| ngEO-SUB-117-WEBC-FUN | Multiple searches | V1 | Search |
| ngEO-SUB-118-WEBC-FUN | Multiple Dataset selection service | V1 | Search |
| ngEO-SUB-119-WEBC-FUN | Interferometric search | V1 | Search |
| ngEO-SUB-121-WEBC-FUN | Catalogue search | V1 | Search |
| ngEO-SUB-122-WEBC-FUN | Search parameters | V1 | Search |
| ngEO-SUB-123-WEBC-FUN | Search parameters | V1 | Search |
| ngEO-SUB-124-WEBC-FUN | Catalogue search settings | V1 | Search |
| ngEO-SUB-126-WEBC-FUN | Catalogue search: spatial criteria | V1 | Search/Map |
| ngEO-SUB-128-WEBC-FUN | Catalogue search: time extent | V1 | Search |
| ngEO-SUB-129-WEBC-FUN | Catalogue search: time slider | V1 | Search |
| ngEO-SUB-130-WEBC-FUN | Catalogue search: download options | V1 | Search |
| ngEO-SUB-131-WEBC-FUN | Catalogue search: Opensearch query | V1 | Search |
| ngEO-SUB-132-WEBC-FUN | Catalogue search: Opensearch query | V1 | Search |
| ngEO-SUB-133-WEBC-FUN | Catalogue search: Use a pre-built Opensearch query | V1 | Search |
| ngEO-SUB-134-WEBC-FUN | Correlation Search Activation | V1 | Search |
| ngEO-SUB-135-WEBC-FUN | Correlation and Interferometric Search Criteria | V1 | Search |
| ngEO-SUB-137-WEBC-FUN | Search results: pagination | V1 | SearchResult |
| ngEO-SUB-138-WEBC-FUN | Search results: pagination | V1 | SearchResult |
| ngEO-SUB-139-WEBC-FUN | Search results: result table | V1 | SearchResult |
| ngEO-SUB-140-WEBC-FUN | Search results: result table | V1 | SearchResult |
| ngEO-SUB-141-WEBC-FUN | Search results: result footprints | V1 | SearchResult/Map |
| ngEO-SUB-142-WEBC-FUN | Search results: products group in search result | V1 | SearchResult |
| ngEO-SUB-143-WEBC-FUN | Search results: Browse images | V1 | SearchResult/Map |
| ngEO-SUB-144-WEBC-FUN | Search results: Browse images | V1 | SearchResult/Map |
| ngEO-SUB-145-WEBC-FUN | GANTT chart | V1 | SearchResult |
| ngEO-SUB-146-WEBC-FUN | GANTT chart | V1 | SearchResult |
| ngEO-SUB-147-WEBC-FUN | GANTT chart | V1 | SearchResult |
| ngEO-SUB-148-WEBC-FUN | Download options edition | V1 | SearchResult |
| ngEO-SUB-149-WEBC-FUN | Use of the search results | V1 | SearchResult |
| ngEO-SUB-150-WEBC-FUN | Search results: Transfer to shopcart | V2 | SearchResult/Shopcart |
| ngEO-SUB-152-WEBC-FUN | Search results: Data access request | V1 | SearchResult/DataAccess |
| ngEO-SUB-153-WEBC-FUN | Search results: Data access request | V1 | SearchResult/DataAccess |
| ngEO-SUB-155-WEBC-FUN | Search results: Planned products' exception | V1 | SearchResult/DataAccess |
| ngEO-SUB-156-WEBC-FUN | My Account: Shopcarts management | V2 | Account/Shopcart |
| ngEO-SUB-157-WEBC-FUN | My Account: Data access requests management | V1 | Account/DataAccess |
| ngEO-SUB-158-WEBC-FUN | Product direct download | V1 | SearchResult |
| ngEO-SUB-159-WEBC-FUN | Data access request types | V1 | DataAccess |
| ngEO-SUB-160-WEBC-FUN | Simple data access | V1 | DataAccess |
| ngEO-SUB-161-WEBC-FUN | Enhanced data access | V2 | DataAccess |
| ngEO-SUB-162-WEBC-FUN | Special Download Option Values | V1 | DataAccess |
| ngEO-SUB-163-WEBC-FUN | Standing order | V2 | DataAccess |
| ngEO-SUB-164-WEBC-FUN | Standing order Types | V2 | DataAccess |
| ngEO-SUB-165-WEBC-FUN | Standing order Pause/Resume | V2 | DataAccess |
| ngEO-SUB-166-WEBC-FUN | Standing order with processing | V2 | DataAccess |
| ngEO-SUB-167-WEBC-FUN | Data Access Requests and Standing Orders monitoring | V2 | DataAccess |
| ngEO-SUB-168-WEBC-FUN | My Account Handling | V1 | Account |
| ngEO-SUB-169-WEBC-FUN | Assignment of a Data access request | V1 | DataAccess |
| ngEO-SUB-170-WEBC-FUN | Estimated download Size | V1 | DataAccess |
| ngEO-SUB-175-WEBC-FUN | My Account functions | V1 | Account |
| ngEO-SUB-176-WEBC-FUN | Sharing of shopcarts | V2 | Shopcart |
| ngEO-SUB-177-WEBC-FUN | Sharing of interface context | V1 | User Interface |
| ngEO-SUB-179-WEBC-FUN | User Authorizations | V2 | Account |
| ngEO-SUB-180-WEBC-FUN | User Categories | V1 | Account |
| ngEO-SUB-181-WEBC-FUN | Inquiries | V1 | Account |
| ngEO-SUB-182-WEBC-FUN | Inquiries Types | V1 | Account |
| ngEO-SUB-183-WEBC-FUN | Help | V1 | User Interface |
| ngEO-SUB-184-WEBC-FUN | Help chapters | V1 | User Interface |
| ngEO-SUB-185-WEBC-FUN | Button description | V1 | User Interface/Tipsy |
| ngEO-SUB-186-WEBC-FUN | DataSet Description | V1 | Search/Tipsy |
| ngEO-SUB-187-WEBC-FUN | Useful links | V1 | User Interface |
| ngEO-SUB-188-WEBC-FUN | 3D Browse/ Vertical Curtain | V1 | Map |
| ngEO-SUB-189-WEBC-FUN | Dataset selection by auto completion | V1 | Search/jQuery Autocomplete |
| ngEO-SUB-201-WEBC-OPE | Browsers' accessibility | V1 | All |
| ngEO-SUB-202-WEBC-OPE | OS accessibility | V1 | All |
| ngEO-SUB-203-WEBC-OPE | GUI customisation | V1 | User Interface |
| ngEO-SUB-204-WEBC-OPE | Plug-in | V1 | Map |
| ngEO-SUB-205-WEBC-OPE | Date | V1 | All |
| ngEO-SUB-206-WEBC-OPE | Configuration | V1 | Configuration |
| ngEO-SUB-207-WEBC-OPE | URLs length | V1 | All |
| ngEO-SUB-215-WEBC-OPE | "Skinnable" GUI | V1 | User Interface |
| ngEO-SUB-301-WEBC-INT | Web server | V1 | All |
| ngEO-SUB-302-WEBC-INT | Product facility | V1 | DataAccess |
| ngEO-SUB-303-WEBC-INT | WMTS interface | V1 | Map |
| ngEO-SUB-304-WEBC-INT | WFS interface | V1 | Map |
| ngEO-SUB-305-WEBC-INT | Web services interface | V1 | Map |
| ngEO-SUB-306-WEBC-INT | Device's local file system | V1 | Map |
| ngEO-SUB-401-WEBC-RAM | Operational life | V1 | All |
| ngEO-SUB-402-WEBC-RAM | Availability | V1 | All |
| ngEO-SUB-701-WEBC-PER | Response time | V1 | All |
| ngEO-SUB-703-WEBC-PER | Datasets | V1 | Search |
| ngEO-SUB-704-WEBC-PER | Shopcarts | V1 | Shopcart |
| ngEO-SUB-705-WEBC-PER | Web Client loading | V1 | All |
| ngEO-SUB-800-WEBC-TVA | Web Server simulator | V1 | Simulator |

## Components to Requirements

|  |  |
| --- | --- |
| **Component** | **Requirement ID(s)** |
| UserInterface | ngEO-SUB-001-WEBC-DES, ngEO-SUB-002-WEBC-DES, ngEO-SUB-004-WEBC-DES, ngEO-SUB-005-WEBC-DES, ngEO-SUB-008-WEBC-DES, ngEO-SUB-013-WEBC-DES, ngEO-SUB-015-WEBC-DES, ngEO-SUB-024-WEBC-DES, ngEO-SUB-025-WEBC-DES, ngEO-SUB-101-WEBC-FUN, ngEO-SUB-102-WEBC-FUN, ngEO-SUB-103-WEBC-FUN, ngEO-SUB-177-WEBC-FUN, ngEO-SUB-183-WEBC-FUN, ngEO-SUB-184-WEBC-FUN, ngEO-SUB-185-WEBC-FUN, ngEO-SUB-187-WEBC-FUN, ngEO-SUB-203-WEBC-OPE, ngEO-SUB-215-WEBC-OPE |
| Map | ngEO-SUB-022-WEBC-DES, ngEO-SUB-023-WEBC-DES, ngEO-SUB-030-WEBC-DES, ngEO-SUB-031-WEBC-DES, ngEO-SUB-033-WEBC-DES, ngEO-SUB-034-WEBC-DES, ngEO-SUB-035-WEBC-DES, ngEO-SUB-036-WEBC-DES, ngEO-SUB-037-WEBC-DES, ngEO-SUB-061-WEBC-DES, ngEO-SUB-104-WEBC-FUN, ngEO-SUB-105-WEBC-FUN, ngEO-SUB-106-WEBC-FUN, ngEO-SUB-107-WEBC-FUN, ngEO-SUB-108-WEBC-FUN, ngEO-SUB-109-WEBC-FUN, ngEO-SUB-110-WEBC-FUN, ngEO-SUB-115-WEBC-FUN, ngEO-SUB-188-WEBC-FUN, ngEO-SUB-204-WEBC-OPE, ngEO-SUB-303-WEBC-INT, ngEO-SUB-304-WEBC-INT, ngEO-SUB-305-WEBC-INT, ngEO-SUB-306-WEBC-INT |
| Search | ngEO-SUB-014-WEBC-DES, ngEO-SUB-048-WEBC-DES, ngEO-SUB-049-WEBC-DES, ngEO-SUB-050-WEBC-DES, ngEO-SUB-058-WEBC-DES, ngEO-SUB-060-WEBC-DES, ngEO-SUB-111-WEBC-FUN, ngEO-SUB-112-WEBC-FUN, ngEO-SUB-113-WEBC-FUN, ngEO-SUB-114-WEBC-FUN, ngEO-SUB-116-WEBC-FUN, ngEO-SUB-117-WEBC-FUN, ngEO-SUB-118-WEBC-FUN, ngEO-SUB-119-WEBC-FUN, ngEO-SUB-121-WEBC-FUN, ngEO-SUB-122-WEBC-FUN, ngEO-SUB-123-WEBC-FUN, ngEO-SUB-124-WEBC-FUN, ngEO-SUB-126-WEBC-FUN, ngEO-SUB-128-WEBC-FUN, ngEO-SUB-129-WEBC-FUN, ngEO-SUB-130-WEBC-FUN, ngEO-SUB-131-WEBC-FUN, ngEO-SUB-132-WEBC-FUN, ngEO-SUB-133-WEBC-FUN, ngEO-SUB-134-WEBC-FUN, ngEO-SUB-135-WEBC-FUN, ngEO-SUB-186-WEBC-FUN, ngEO-SUB-189-WEBC-FUN, ngEO-SUB-703-WEBC-PER |
| SearchResult | ngEO-SUB-028-WEBC-DES, ngEO-SUB-043-WEBC-DES, ngEO-SUB-045-WEBC-DES, ngEO-SUB-046-WEBC-DES, ngEO-SUB-137-WEBC-FUN, ngEO-SUB-138-WEBC-FUN, ngEO-SUB-139-WEBC-FUN, ngEO-SUB-140-WEBC-FUN, ngEO-SUB-141-WEBC-FUN, ngEO-SUB-142-WEBC-FUN, ngEO-SUB-143-WEBC-FUN, ngEO-SUB-144-WEBC-FUN, ngEO-SUB-145-WEBC-FUN, ngEO-SUB-146-WEBC-FUN, ngEO-SUB-147-WEBC-FUN, ngEO-SUB-148-WEBC-FUN, ngEO-SUB-149-WEBC-FUN, ngEO-SUB-150-WEBC-FUN, ngEO-SUB-152-WEBC-FUN, ngEO-SUB-153-WEBC-FUN, ngEO-SUB-155-WEBC-FUN, ngEO-SUB-158-WEBC-FUN |
| Account | ngEO-SUB-156-WEBC-FUN, ngEO-SUB-157-WEBC-FUN, ngEO-SUB-168-WEBC-FUN, ngEO-SUB-175-WEBC-FUN, ngEO-SUB-179-WEBC-FUN, ngEO-SUB-180-WEBC-FUN, ngEO-SUB-181-WEBC-FUN, ngEO-SUB-182-WEBC-FUN, |
| DataAccess | ngEO-SUB-038-WEBC-DES, ngEO-SUB-152-WEBC-FUN, ngEO-SUB-153-WEBC-FUN, ngEO-SUB-155-WEBC-FUN, ngEO-SUB-157-WEBC-FUN, ngEO-SUB-159-WEBC-FUN, ngEO-SUB-160-WEBC-FUN, ngEO-SUB-161-WEBC-FUN, ngEO-SUB-162-WEBC-FUN, ngEO-SUB-163-WEBC-FUN, ngEO-SUB-164-WEBC-FUN, ngEO-SUB-165-WEBC-FUN, ngEO-SUB-166-WEBC-FUN, ngEO-SUB-167-WEBC-FUN, ngEO-SUB-169-WEBC-FUN, ngEO-SUB-170-WEBC-FUN, ngEO-SUB-302-WEBC-INT |
| Shopcart | ngEO-SUB-039-WEBC-DES, ngEO-SUB-040-WEBC-DES, ngEO-SUB-041-WEBC-DES, ngEO-SUB-042-WEBC-DES, ngEO-SUB-062-WEBC-DES, ngEO-SUB-150-WEBC-FUN, ngEO-SUB-156-WEBC-FUN, ngEO-SUB-176-WEBC-FUN, ngEO-SUB-704-WEBC-PER, |
| Configuration | ngEO-SUB-206-WEBC-OPE |
| Simulator | ngEO-SUB-800-WEBC-TVA |

END OF DOCUMENT