



---

# GIT Barents Internet Service



• • • • • • • • • •

# GIT Barents Architecture Specification

*Description of the Internet  
Infrastructure and Webb client*

Version 1.0

---



---

# Content

<b>Executive Summary</b> .....	<b>4</b>
<b>Introduction</b> .....	<b>5</b>
<b>OGC Specifications used</b> .....	<b>6</b>
Web Map Service (WMS) .....	6
Web Feature Service (WFS).....	7
Web Map Context (WMC) .....	7
Styled Layer Descriptor (SLD) .....	7
Geography Markup Language (GML) .....	8
<b>GITBarents project</b> .....	<b>9</b>
Layered architecture .....	9
Data layer.....	9
Processing layer .....	11
Client layer .....	11
GITBarents Viewer Configuration.....	11
Mapping Configuration, NMA in Kiruna, Sweden .....	12
Auxiliary equipment used in NMA, Kiruna .....	13
Portrayal of GBDB .....	13
Default portrayal .....	13
Alternative portrayal .....	13
Metadata.....	13
Metadata standard and implementation .....	13
Metadata access.....	13
Metadata presentation.....	14
<b>References</b> .....	<b>15</b>
<b>GITBarents Client Functionality</b> .....	<b>16</b>
Description.....	16
Manage Context (WMC).....	16
Functionality .....	16
Implementation - Save context.....	16
Implementation - Load context .....	16
Search function (WFS).....	17
Functionality .....	17
Implementation .....	17
Set style (SLD) .....	17
Functionality .....	17
Implementation .....	17
Add external service .....	18
Functionality .....	18
Implementation .....	18
GetCapability and GetFeatureInfo Request (WMS) .....	19
Functionality GetCapability .....	19
Functionality GetFeatureInfo .....	19
Implementation .....	19
<b>Documentation (JavaDoc)</b> .....	<b>19</b>
JavaScript – documentation available from project manager .....	19
JavaServlets – documentation available from project manager .....	19
Java Server Pages – documentation available from project manager.....	19
Tutorial – documentation available from project manager .....	19
<b>Screenshots of GITBarents client</b> .....	<b>20</b>





## Executive Summary

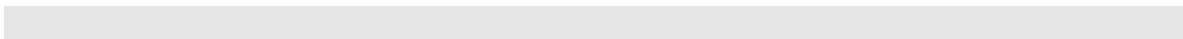
The WMS/WFS technology enables to combine into one view geospatial and thematic information, maintained by several independent service providers. The Barents GIT II Project has used this technique to establish an Internet-based infrastructure, where both maps and thematic information are presented simultaneously for the users in WMS/WFS applications. The geographic part of the data is fetched from servers at the national mapping agencies of Finland, Norway and Sweden, and – within short – from a server at the Regional Administration of Murmansk oblast in Russia.

The basic ideas behind these techniques are to provide geographical information to a user community in a standardized manner in order to facilitate the exchange of information independent on hard- and software. The other important issue that is better supported using these standards, is that the information is provided from the source where it is maintained and kept updated. This way it will minimize the overhead of regularly requiring updated data from a provider, and import of this data into another system. The interoperability that is achieved by using these web mapping standards also means that it is possible to combine information from several sources in one client, even if there are varying technical implementation in terms of hard- and software, as long as the providing system support the OGC web mapping standards.

There are several ways to implement the system that provide these types of web services. These include purchase of vendor specific WMS software, obtain an available open source package or develop a new map-server based on the available specifications of these services.

It is also important to realize that after setting up these types of web services, only the first step that is the access to the information, is fulfilled. Hereafter it is necessary to provide or advise the user with some interface and tools how to view and interact with required the information. Depending on the demands for the user to interact with the data, there are several options to be considered when deciding what client category to use. If only the most basic view and zoom functions are needed a simple “thin” client based on HTML might be sufficient. A more advanced use of the data in a geographical information system (GIS) on the other hand requires more logic and functionality to be implemented in a “thick” client. Within the GITBarents project a client is developed that can be characterised as an intermediate category. The GITBarents client supports all of the basic tools that are needed for explorations of the database that is constructed within the GITBarents II project. Extending this basic functionality the client also give the user the opportunity to add, view and explore data from other providers of web mapping services. This user defined content of web maps can be saved locally and later reloaded into the client. The rendering of the map can be altered by the user in several ways, by changing the order of which the different layers are rendered on top of each other, or altering the colour by using “styles” defined for each layer.

To obtain information about the data viewed in the client, metadata, the user can request that information independent on the web map service implementation. For the data compiled within or in collaboration to the GITBarents project, the metadata content corresponds to “core metadata” defined in the ISO 19115 standard. More detailed information about a specific object (i.e. a lake, settlement) in the map is supported by the standard request “getFeatureInfo” that is presented in table format.



## Introduction

This document describes the GITBarents System Architecture. As stated in the project proposal, the GITBarents project aims at establishing a standard-based infrastructure through which an Internet user can access the geospatial data of the GITBarents database, hosted at national servers in Norway, Sweden, Finland and Russia.

Several organisations and companies provide systems for online mapping. Many of these solutions are implemented using proprietary systems that lack possibilities to communicate with other implementations. The figures below show the principle difference between a web map systems that doesn't support interoperability (figure1) and one that do (figure 2).

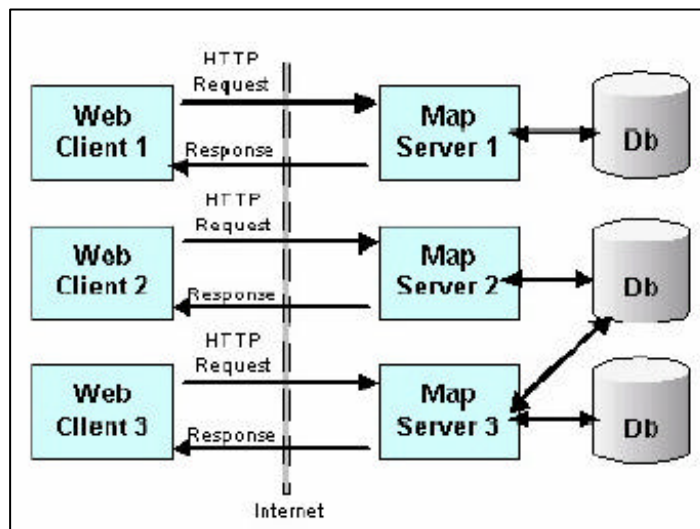


Figure 1. Non-Interoperability in a web mapping client server solution (source OpenGIS Web Map Server Cookbook).

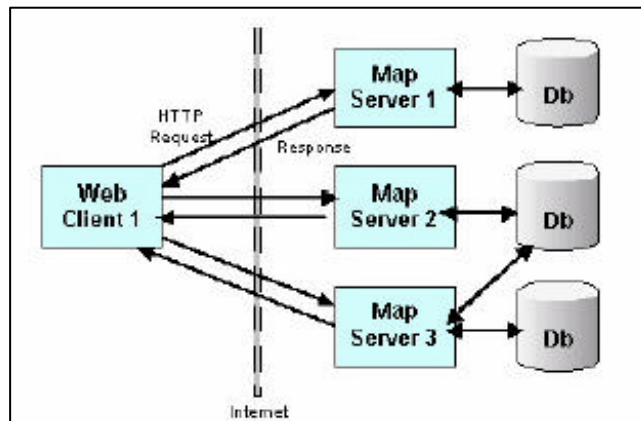


Figure 2. Interoperability in a web mapping client server solution (source OpenGIS Web Map Server Cookbook).

The approach in figure 2 allow the user to run one single client that access the geographic information from several servers and to make use of information from many service providers simultaneously. This approach also support the concept that data should be distributed from the creator or managing organisation of data in order to ensure distribution of updated and accurate information.



## OGC Specifications used

OGC Web Services are an evolutionary, standards-based framework that will enable seamless integration of a variety of online geoprocessing and location services. OGC Web Services will allow distributed geoprocessing systems to communicate with each other using technologies such as XML and HTTP. This means that systems capable of working with XML and HTTP will be able to both advertise and use OGC Web Services.

OGC Web Services will allow future applications to be assembled from multiple, network-enabled geoprocessing and location services. This capability will be possible because rules will be established for these services to advertise the functionality they provide and how to send service requests via open, standard methods. In this manner, OGC Web Services will provide a vendor-neutral interoperable framework for web-based discovery, access, integration, analysis, exploitation and visualization of multiple online geodata sources, sensor-derived information, and geoprocessing capabilities.

The OGC Web Services (OWS) suite includes three principal types of georeferenced information access services. Besides WMS, it also includes the Web Coverage Server (WCS) and the Web Feature Server (WFS). Other standards include the Simple Feature Specification (SFS), the Geography Markup Language (GML), and others. While these standards are independent of each other, they are complementary to each other.

### Web Map Service (WMS)

The WMS specification standardizes three operations that are used to request a map from a client application.

- GetCapabilities

The purpose of the GetCapabilities request is to declare the GetMap services that you provide. You must be able to deliver an XML metadata file via http upon receiving a request such as:

```
http://www.myserver.com/wms/process.cgi?  
REQUEST=GetCapabilities&VERSION=1.1.1&SERVICE=WMS
```

The response from such request will be a XML document that contains all the information necessary to build the requests for obtaining more information and the map image.

- GetMap

A WMS server must be able to deliver a map via http upon receiving a client request such as:

```
http://www.myserver.com/wms/process.cgi?  
REQUEST=GetMap&FORMAT=image/gif&WIDTH=640&HEIGHT=480&  
LAYERS=temperature&SRS=EPSG:4326&  
BBOX=-110.,40.,-80.,30.&VERSION=1.1.1
```

The response from such request will be a raster image of the database and not the actual data.

- GetFeatureInfo

This option enables you to provide further information about a specific pixel in a previous GetMap request. You may choose to implement this service for any or all layers. The URL request resembles the GetMap request, but the user will also specify a single pixel. A typical request might be:

```
http://yourserver.ext/script.cgi?REQUEST=GetFeatureInfo
&WIDTH=640&HEIGHT=480&BBOX=-110.,40.,-80.,30.
&VERSION=1.1.1&SRS=EPSG:4326&QUERY_LAYERS=temperature
&X=321&Y=165
```

### **Web Feature Service (WFS)**

The OGC Web Map Service allows a client to overlay map images for display served from multiple Web Map Services on the Internet. In a similar fashion, the OGC Web Feature Service allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services.

### **Web Map Context (WMC)**

WMS 1.1.1 specifies how individual map servers describe and provide their map content. The present Context specification states how a specific grouping of one or more maps from one or more map servers can be described in a portable, platform-independent format for storage in a repository or for transmission between clients. This description is known as a "Web Map Context Document," or simply a "Context." Presently, context documents are primarily designed for WMS bindings. However, extensibility is envisioned for binding to other services. A Context document includes information about the server(s) providing layer(s) in the overall map, the bounding box and map projection shared by all the maps, sufficient operational metadata for Client software to reproduce the map, and ancillary metadata used to annotate or describe the maps and their provenance for the benefit of human viewers. A Context document is structured using eXtensible Markup Language (XML). Annex A of this specification contains the XML Schema against which Context XML can be validated. There are several possible uses for Context documents: - The Context document can provide default start-up views for particular classes of user. Such a document would have a long lifetime and public accessibility. - The Context document can save the state of a viewer client as the user navigates and modifies map layers. - The Context document can store not only the current settings but also additional information about each layer (e.g., available styles, formats, SRS, etc.) to avoid having to query the map server again once the user has selected a layer. - The Context document could be saved from one client session and transferred to a different client application to start up with the same context. Contexts could be catalogued and discovered, thus providing a level of granularity broader than individual layers.

### **Styled Layer Descriptor (SLD)**

This document addresses the need for geospatial consumers (either humans or machines) to control the visual portrayal of the data with which they work. The current OpenGIS Web Map Service (WMS) specification supports the ability for an information provider to specify very basic styling options by advertising a preset collection of visual portrayals for each available data set. However, while a WMS currently can provide the user with a choice of style options, the WMS can only tell the user the name of each style. It cannot tell the user what portrayal will look like on the map. More importantly, the user has no way of defining their own styling rules. The ability for a human or machine client to define these rules requires a styling language that

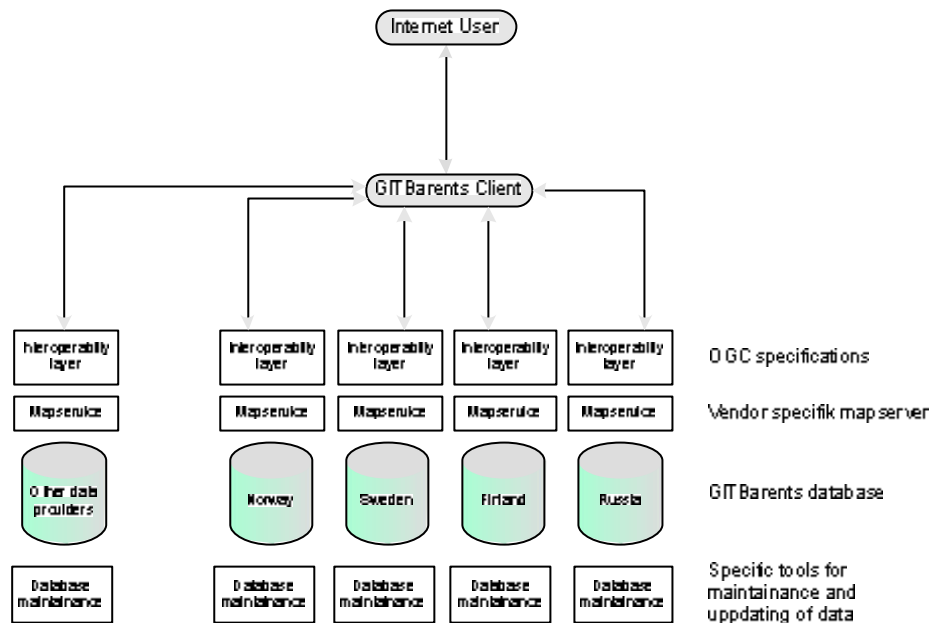


the client and server can both understand. Defining this language, called the *StyledLayerDescriptor (SLD)*, is the main focus of this paper, and it can be used to portray the output of Web Map Servers, Web Feature Servers and Web Coverage Servers. In many cases, however, the client needs some information about the data residing on the remote server before he, she or it can make a sensible request. This led to the definition of new operations for the OGC services [see Section 6.6] in addition to the definition of the styling language. There are two basic ways to style a data set. The simplest one is to colour all features the same way. For example, one can imagine a layer advertised by a WMS as “hydrography” consisting of lines (rivers and streams) and polygons (lakes, ponds, oceans, etc.). A user might want to tell the server to colour the insides of all polygons in a light blue, and colour the boundaries of all polygons and all lines in a darker blue. This type of styling requires no knowledge of the attributes or “feature types” of the underlying data, only a language with which to describe these styles. This requirement is addressed by the **FeatureTypeStyle** element in the SLD document. A more complicated requirement is to style features of the data differently depending on some attribute. For example, in a roads data set, style highways with a three-pixel red line; style four-lane roads in a two-pixel black line; and style two-lane roads in an onepixel black line. Accomplishing this requires the user to be able to find out what attribute of the data set represents the road type. WMS already has an optional operation that fulfils this need, called **DescribeLayer**. This operation returns the feature types of the layer or layers specified in the request, and the attributes can be discovered with the **DescribeFeatureType** operation of a WFS interface.

### **Geography Markup Language (GML)**

The Geography Markup Language (GML) is an XML encoding in compliance with ISO 19118 for the transport and storage of geographic information modelled according to the conceptual modelling framework used in the ISO 19100 series and including both the spatial and non-spatial properties of geographic features.

## GITBarents project



GITBarents Internet access

### Layered architecture

The GITBarents architecture consists of three layers: Data layer, Processing layer and Client layer.

#### Data layer

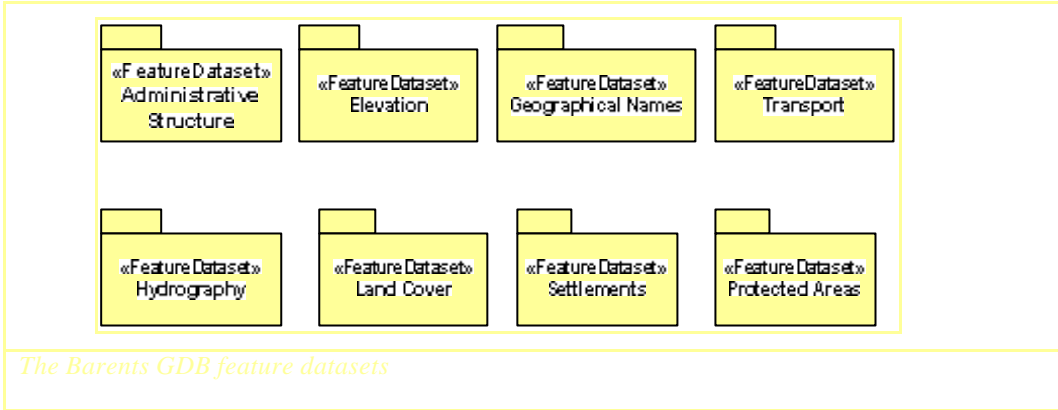
The Data Layer constitutes the content, the geographic information, of the map services provided. In this project represented as the "Barents geographic database, the Barents GDB". The data model and the feature catalogue of the Barents GDB are documented in the "Barents GDB Data Specifications". The database consists of 21 feature classes organized into 8 thematic groups.

The implementation of the data layer is based on ESRI's Geodatabase. The Geodatabase is an object-oriented data model introduced by ESRI that represents geographic features and attributes as objects and the relationships between objects. The geodatabase is hosted inside a relational database management system. The Geodatabase contain feature classes and tables (non-spatial data). Feature classes can be organized into feature datasets.

The Geodatabase implementation of the Barents GDB is designed using UML<sup>1</sup>. The UML model organizes the Barents GDB into 8 feature datasets corresponding to the thematic groups in the data specification.

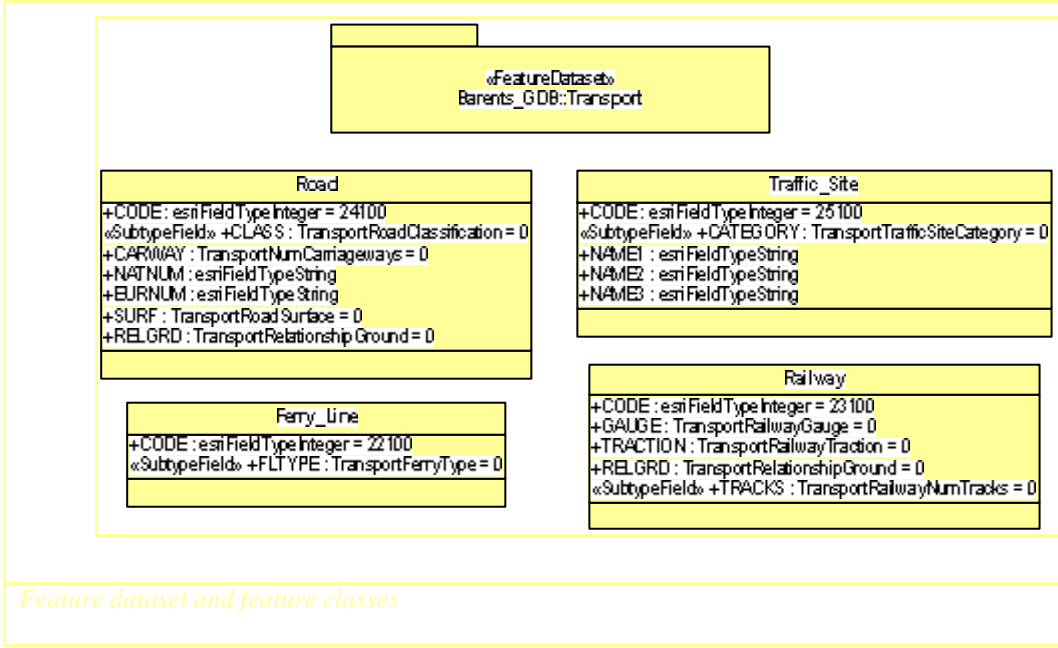
<sup>1</sup> UML, Unified Modelling Language. A graphical language for object modelling

•  
•  
•  
•  
•  
•  
•



*The Barents GDB feature datasets*

Each feature dataset holds the feature classes thematically related the dataset. Figure XX is an example which shows the Road, Ferry\_line, Traffic\_Site and Railway feature classes belonging to the Transportation feature dataset.



*Feature dataset and feature classes*

A complete UML model of the Barents GDB is available from the project manager.

The data layer software platform is Oracle and ArcSDE. The hardware and system configuration details differ between the four countries. The Geodatabase schema of the Barents GDB instance is created using Case tools<sup>2</sup> in a three step process:

1. Design of Barents GDB UML model using Visio 2000.
2. Export of the UML model to a XMI document.
3. Create the Geodatabase schema in Oracle/ArcSDE by using the Schema Wizard in ArcCatalog.

### **Processing layer**

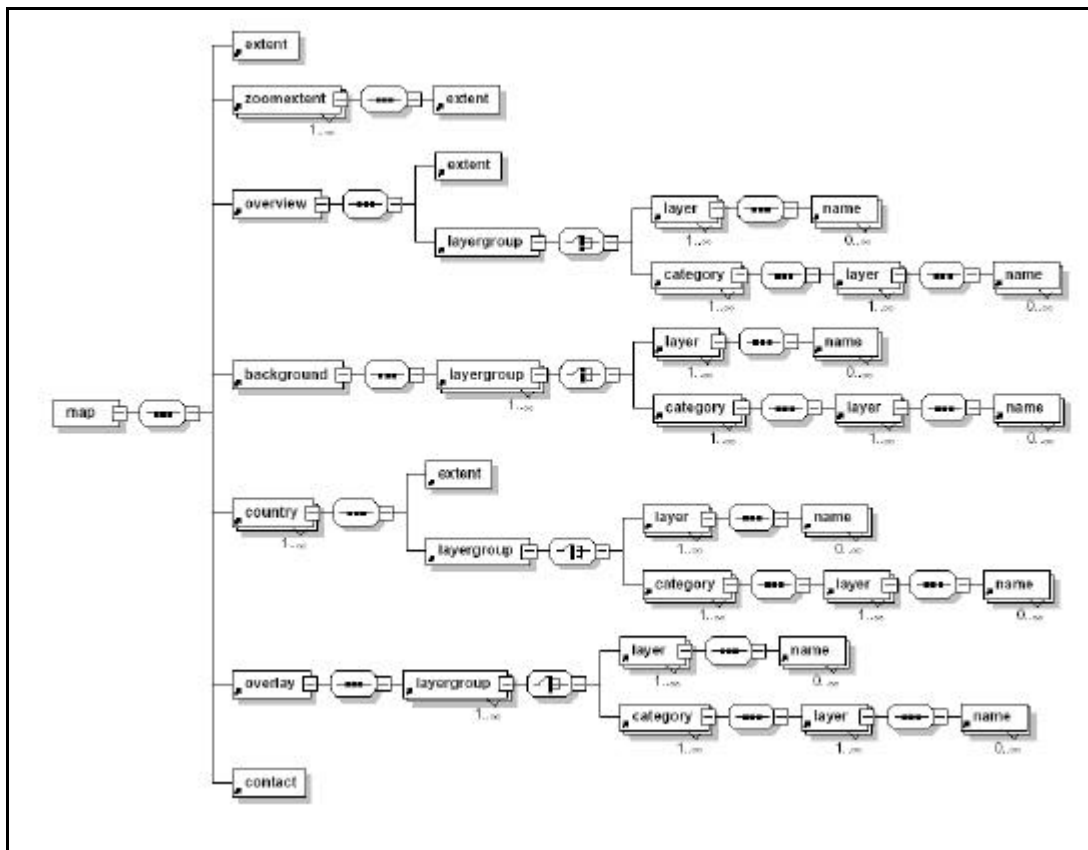
ESRI ArcIMS and the corresponding WMS/WFS connectors.

### **Client layer**

The GITBarents client is based of plain HTML and JavaScript.

Further information and use of the GITBarents client in Appendix A

## **GITBarents Viewer Configuration**



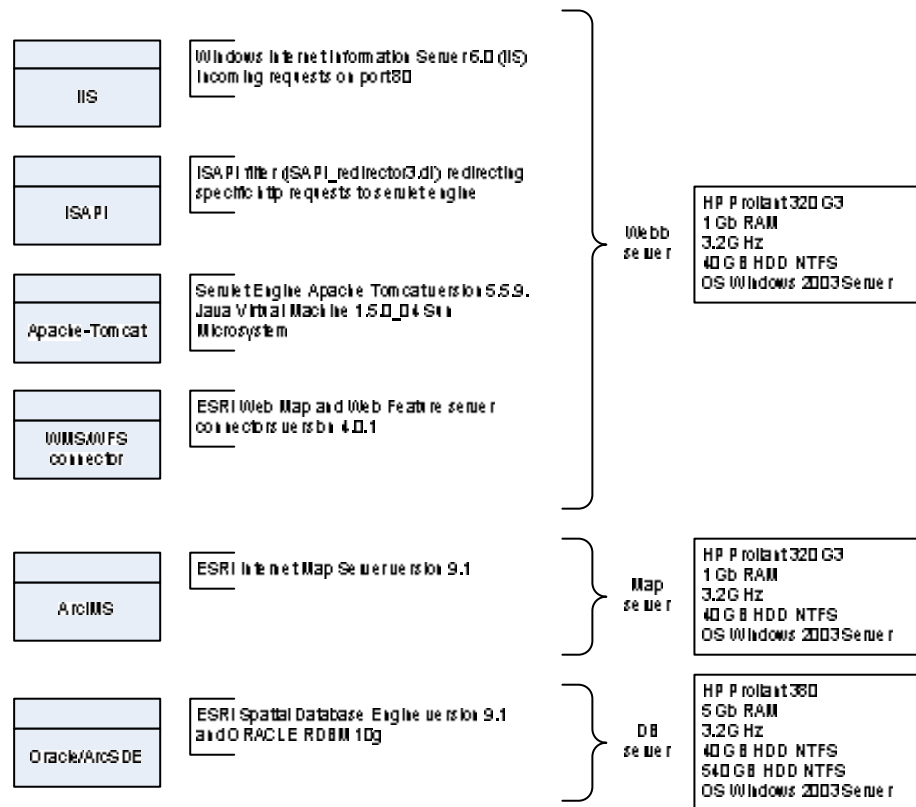
Structure of the [gitbarentsviewer.xml]

<sup>2</sup> CASE, Computer-aided software engineering. A category of software that provides a development environment for programming teams. CASE systems offer tools to automate, manage and simplify the development process.

## Mapping Configuration, NMA in Kiruna, Sweden

A general setup of the mapping system used in The GITBarents project is not possible to describe since the participating organisations may have other requirements on the system or conditions that has to be fulfilled. To accomplish the goals of the GITBarents project the requirement on each NMA is to supply the geographical information, of the database, according to OGC specifications of WMS and WFS services.

The figure below show the specific configuration of the Internet Mapping System in Sweden, Kiruna as on 2005.



Specific setup of the Web Mapping System used in the GITBarents project at the National Land Survey in Kiruna.

The Jakarta ISAPI filter is a replaceable dynamic link library (DLL) the server calls on every HTTP (Hypertext Transfer Protocol) request. When the filter is loaded it tells the server what sort of notifications it is interested in. After that, whenever the selected events occur, the filter is called and given the opportunity to process that event. In the configuration file to this filter the URI:s to "servlet", "wmsconnector" and "wfsconnector" are specified as events that are redirected to the servlet container instead of IIS.

The WMS and WFS connectors used take care of the requests that conform to the OGC specifications and transfer the requested command to the mapping server. These components, the connectors and mapping server, may be of any manufacture compliant to the WMS/WFS specifications. There are a number of non-commercial WMS/WFS servers available e.g. Geoserver or Mapserver that are commonly used.

The storage of the GITBarents database is also a choice to be made based on economical conditions and demand on the system as scalability and performance. Data might be stored as files, commonly shape files, in a file system or in any other spatially enabled database.

#### ***Auxiliary equipment used in NMA, Kiruna***

The backup system is an EMC Legato NetWorker 7 backup system connected to an ADIC Scalar i2000 Fibre Channel LTO-2 Tape Library. NetWorker Module for Oracle and NetWorker Module for SQL Server is used to backup the database servers. Backup is done on daily basis with one full backup every week.

The power infrastructure consists of several Uninterruptible Power Systems (UPS), one for each rack. A Power Manager System safely shuts down all systems in case of a longer power failure.

The web server is located on the public network outside the firewall. Only permitted requests of HTTP and HTTPS are passing through from the outside. The map and database server are located on the internal network.

### **Portrayal of BGDB**

The rendering of geographic databases is heavily dependent on scale.

BGDB is available in generalisations optimized for scale range 1:1M and 1:3M. Above 1:5M the 1:3M database is used and between 1:5M and 1:300 000 the 1:1M database I used. Below 1:300 000 the map will no be rendered.

#### ***Default portrayal***

By default the use of the GITBarents WMS services will use settings applied in the mapservice setting defined in AXL files used by the ESRI ArcIMS software. This setup will render a map over the Barents region similar to the Figure A.1 in appendix A.

#### ***Alternative portrayal***

The use of styles allow for interactive change of the portrayal of the geographic information.

### **Metadata**

The metadata of the GITBarents database is compiled from data that were available from the National Mapping Agencies.

#### ***Metadata standard and implementation***

The metadata follow the ISO19115 standard and is implemented as XML documents produced by ESRI ArcCatalogue metadata tool.

#### ***Metadata access***

Each layer in the database has a related XML file of metadata. The location of metadata files are defined by the capabilities xml-file of the WMS service.

```
<MetadataURL type="ISO">
<Format>text/xml</Format>
<OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="http://webserver/metadata/gitbarents/layer.xml"
xlink:type="simple" />
</MetadataURL>
```



### Metadata presentation

The GITBarents viewer implements a transformation of the metadata xml-file using XSL that convert the xml document into HTML.

## Metadata - Forest (se)

**View options - Metadata**

FOREST\_1M ▾

- Overview
- Detailed information
  - Description
  - Quality
  - Contact
  - Geographical
  - Distribution
- XML

**Overview**

**Title**  
Forest

**Date**  
20011231, creation  
20051231, revision  
20051231, publication

**Edition**  
Version 2005 (20051231)

**Presentation Form**  
digital map

**Language**  
English, Swedish

**Character Set**  
utf8 - 8 bit UCS Transfer Format

**Topic Category**  
biota

**Keywords**  
*Theme keywords:* forest  
*Place keywords:* Europe, Barents region, Sweden, Norrbotten county, Västerbotten county

**Spatial Representation Type**  
vector

---

Barents GIT II project, 20052006-02-08

## References

**OGC cookbook**

URL: <http://www.opengeospatial.org/resources/?page=cookbooks>

**OGC specification WMS service**

URL: <http://www.opengeospatial.org/specs/?page=specs>

**OGC specification WFS service**

URL: <http://www.opengeospatial.org/specs/?page=specs>

**OGC specification WMC service**

URL: <http://www.opengeospatial.org/specs/?page=specs>

**OGC specification SLD service**

URL: <http://www.opengeospatial.org/specs/?page=specs>

**OGC specification GML service**

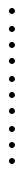
URL: <http://www.opengeospatial.org/specs/?page=specs>

**GIModig project description**

URL: <http://gimodig.fgi.fi/deliverables.php>

**INSPIRE Architecture & Standards Position Paper**

URL: <http://www.ec-gis.org/inspire>



## GIT Barents Client Functionality

### Description

The default client of the GITBarents database is accessed from the homepage. The client is compatible with the IE6, Mozilla Firefox, Opera and NetScape 7.0 Web Browser.

The client is by default using a setup that allow the user to make each layer of the GITBarents database visible or hidden in order to allow for individual choices of layers to be portrayed.

At the bottom a menu is available to select some basic functionalities. From this menu more detailed selections are available from the “Menu” selection.

### Manage Context (WMC)

#### Functionality

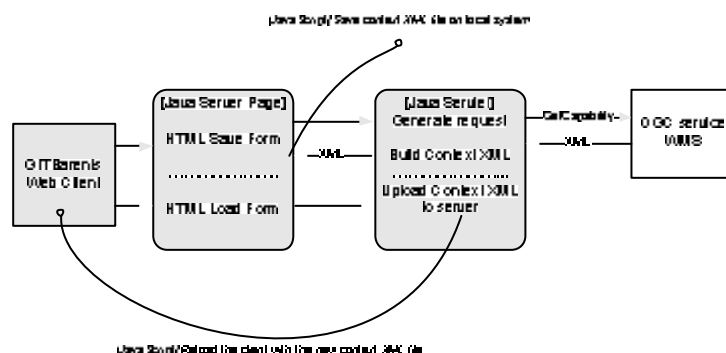
The “Manage context” function is divided in to two different sub functions. The first one is a save context function, that makes it possible to **save** the current configuration of the client. The second sub function provide the possibility to **load** a previously saved context into the client

#### Implementation - Save context

1. The ManageContext function runs from a JSP page. When the user selects the saveContext button the JSP page uses the JavaScript API to create the context document. And the context document is sent to a servlet.
2. The servlet receives the context document and uses it to fill in more information from each services. When the context document is completed with all required information the servlet returns it to the client and the user can save it locally for later use.

#### Implementation - Load context

1. The ManageContext function runs from a JSP page. When the user selects the load context button, a context.xml file is uploaded to the server by a servlet. The servlet will write the unique file to disc and redirect the name and location to a JSP page.
2. The JSP page will process the uploaded context and use the JavaScript API to load the context information in to the client.



## Search function (WFS)

### Functionality

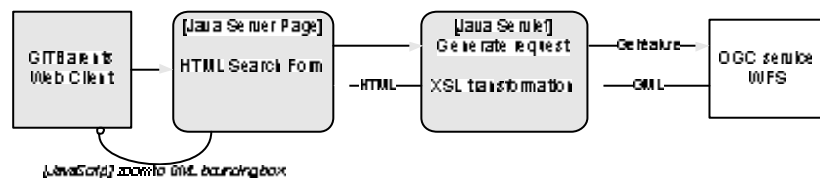
Allow user to make a feature search in the GITBarents database. Not all layers are searchable. Searchable layers are for example:

- Settlement area/point
- Water body
- Protected Area area/point
- Traffic site

### Implementation

The search function is using the Web Feature Sever (WFS).

1. The search function runs from a JSP popup. The JSP uses the JavaScript API to fill the menu with searchable services and layers. The service and layer the user has selected is sent to a servlet.
2. The servlet is sending a WFS getFeature request to the service, to list the objects in the feature class. The response is a GML XML file that is processed by a XSL processor and returned to the popup as an HTML document.
3. The user can select between objects in the requested feature class. When an object is selected, a request is sent back to the servlet.
4. The servlet is sending a new WFS getFeature request to the service, asking for the GML of the selected object. The GML bounding box is sent back to the JSP popup.
5. The popup will use the JavaScript API to Zoom to the selected feature.



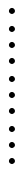
## Set style (SLD)

### Functionality

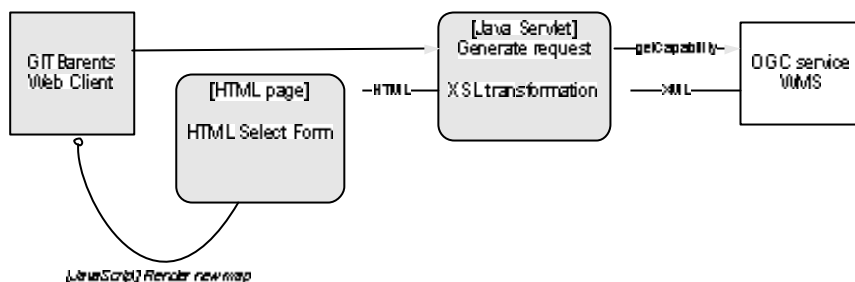
Set style is a function that allows the user to switch style on a single layer. The user can select between a number of predefined styles.

### Implementation

1. When the user click on a layer in the layer list, a request with layer name and the URL to that service is sent to a servlet.
2. The servlet sends a getCapability request to the service, and get a capability XML document as the response.



3. The XML document is processed by an XSL processor and returned to the user as a HTML document. From this document, the user can select between a number of styles.
4. When the user selects a style the HTML document is using the JavaScript API to update the client.



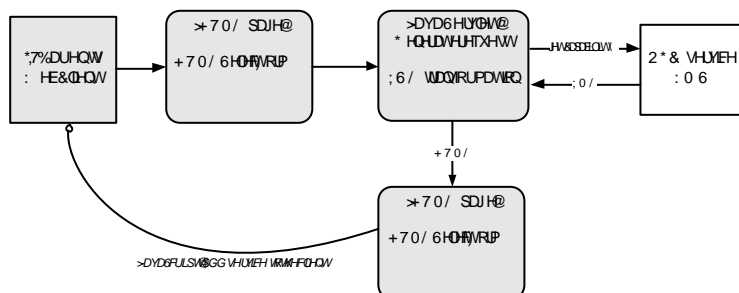
## Add external service

### Functionality

This function allows the user to add layers from other WMS services. If the requested service support the SLD (Styled Layer Descriptor), the style used to render the layer can also be set when adding the layer.

### Implementation

1. HTML form where the user can add a URL to a service, this URL is sent to a servlet.
2. The servlet builds a capability request from the URL, the request is sent to the service. The response is a capability document in XML format. The servlet transforms the XML document in to an HTML document and sends it back to the client.
3. When the user selects a layer and style the action is using the JavaScript API to update the client.



Data flow GITBarents addService function

## GetCapability and GetFeatureInfo Request (WMS)

### **Functionality GetCapability**

Allow the user to get a capability document from a certain service. Document will be returned as HTML document. Optionally the capability document is returned as XML.

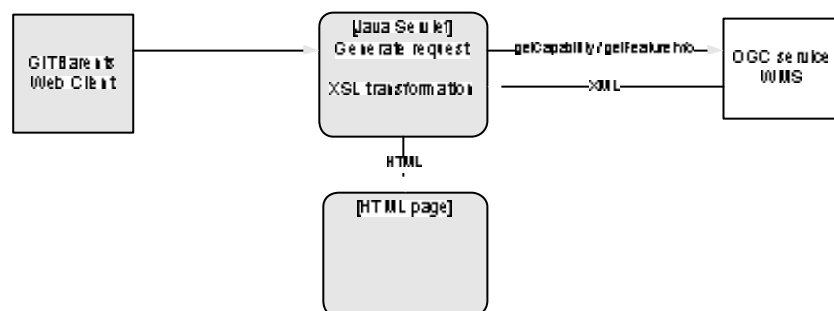
### **Functionality GetFeatureInfo**

Allow the user to click on a certain object in the map and get specific information on that object.

### **Implementation**

Figure 2. shows a schematic view of the GetCapability/GetFeatureInfo Request.

1. The request with the URL to the selected service is sent to a servlet.
2. The servlet will send either a GetCapability or GetFeatureInfo request to the service. The XML response is returned to the servlet.
3. The response is processed with XSL and returned as a HTML document.



## Documentation (JavaDoc)

**JavaScript** – documentation available from project Co-ordinator

**JavaServlets** – documentation available from project Co-ordinator

**Java Server Pages** – documentation available from project Co-ordinator

**Tutorial** – documentation available from project Co-ordinator

Screenshots of the GIT Barents client

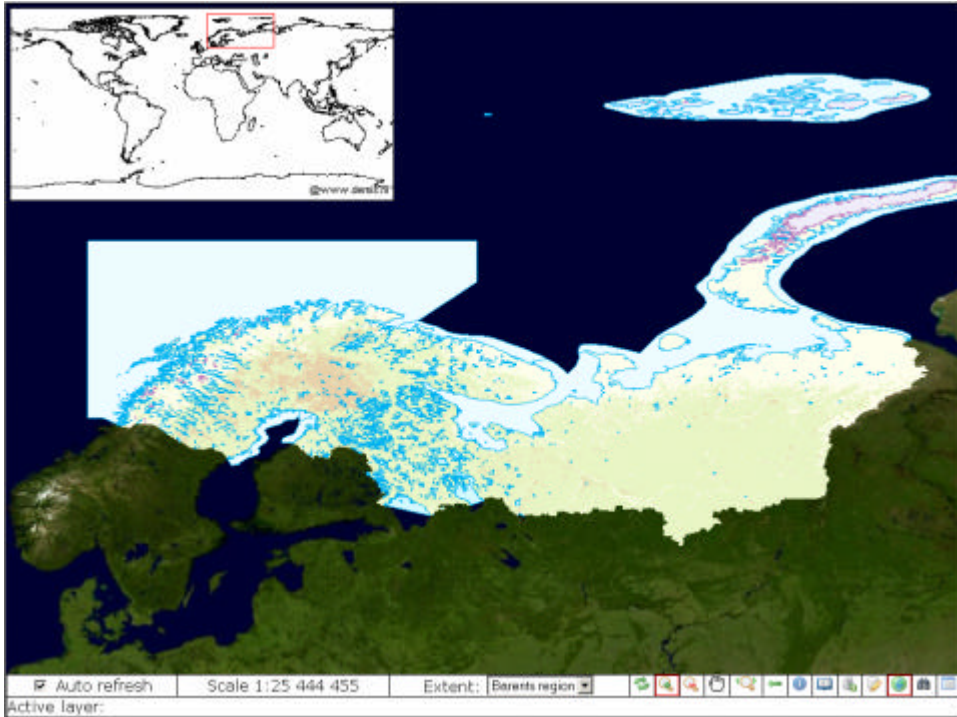


Figure A1. Default startup extent and portrayal of the GITBarents database.

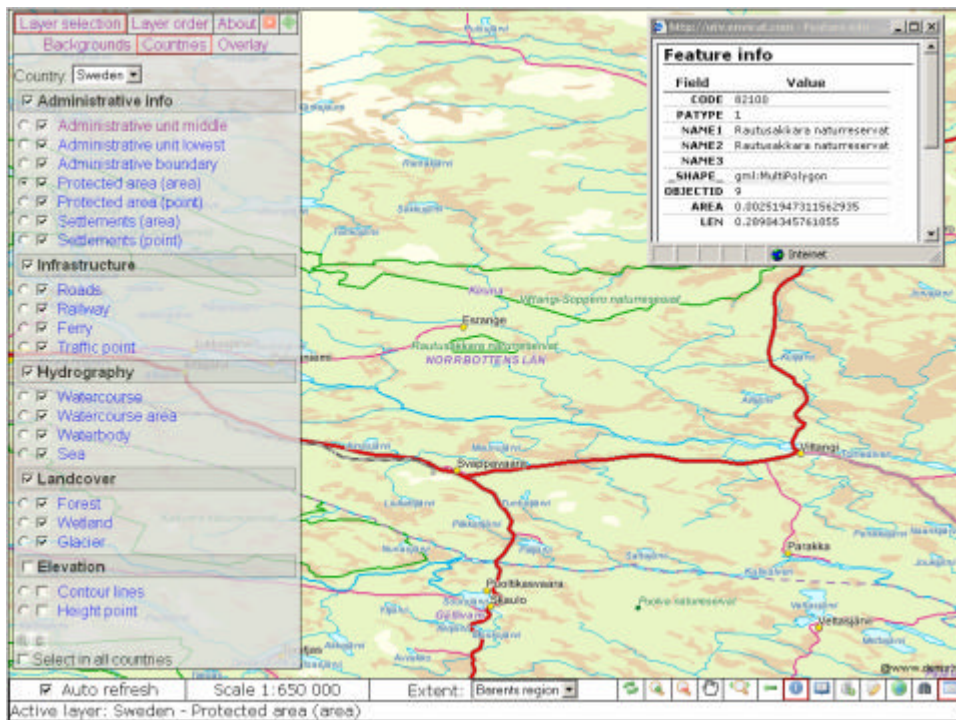


Figure A2. Request of feature information

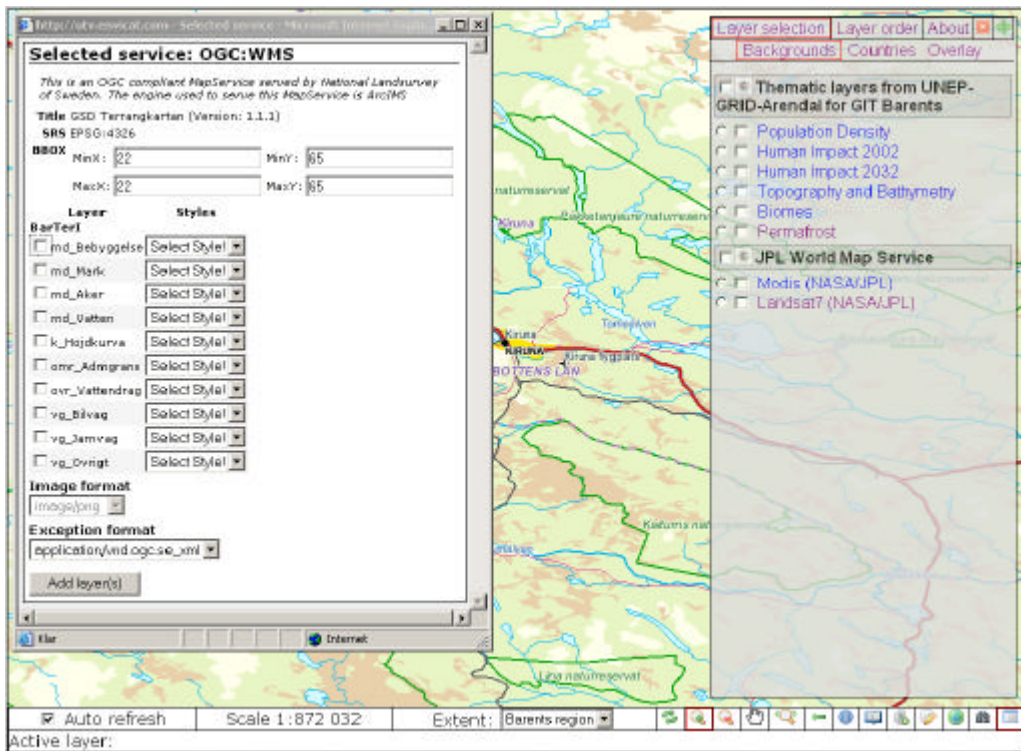


Figure A3. Possibility to interactively add new Web Map Services to the client

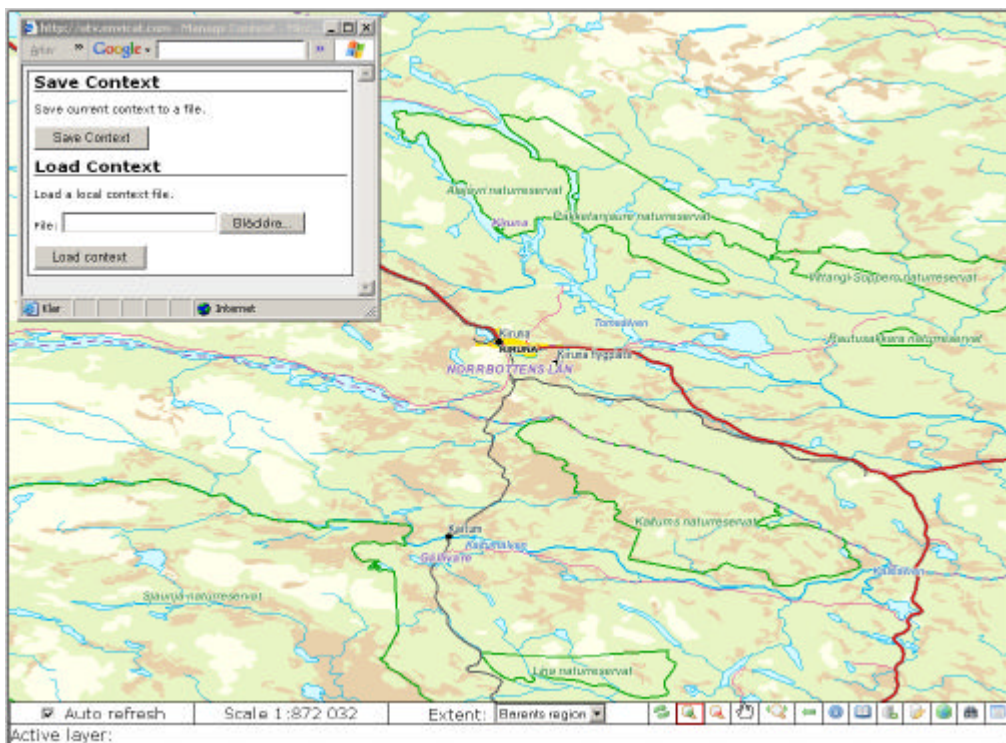


Figure A4. Possibility to interactively locally save and load a context file, describing the current client configuration.