Retrieval, caching, manipulation and visualization of GIS data in Location Aware Applications

MAGNUS NILSSON
PER NILSSON

MASTER OF SCIENCE PROGRAMME

Department of Environmental Engineering
Division of Geographic Information Technology
Retrieval, caching, manipulation and visualization of GIS data in Location Aware Applications

By
Magnus Nilsson
Per Nilsson

Division of Geographic Information Technology
Department of Environmental Engineering
Luleå University of Technology

2002-06-14
Abstract

This thesis is made for the Alipes project. Alipes is a project at the Center for Distance-spanning Technology (CDT). CDT is a department at Luleå University of Technology.

The purpose with this thesis is to investigate and evaluate a standard for transferring GIS data to a mobile user and to introduce a new map retrieval system for Location Aware Applications in the Alipes project. Also, a way to lower the cost related to map data, by introducing caching into the system, is investigated. An application that implements these areas has been created.

There is a need for geographic information standards for all businesses that produces, distributes or utilizes spatial information. Since digital data from a wide variety of sources is used by a diversity of applications, a standard for transferring geographic information called Web Map Service (WMS) has been developed by the Open GIS Consortium (OGC).

A problem with the currently net-based map retrieval is that it’s very expensive and therefore prohibitive for usage in Location Aware Applications. That is; there is a need to reduce the number of request to the map server.

A way to solve this problem is to use caching. Caching is a way to store and reuse data. There are different ways of caching data; the most common is to cache it locally (on disc) or to use a proxy server.

An application that uses the WMS standard and caching is developed. It is implemented in a handheld computer (PDA), and retrieves its position and connects to the map server via WLAN. Using WMS also makes it possible to retrieve additional information about objects on the map, such as restaurants, movie theaters etc.

The caching method used in the application is to download a larger map then the one requested. A smaller map is then cut out from the larger map and then used by the application. This way the client doesn’t have to request the server for a new map unless the requested area is outside the downloaded map, hence the number of requests is reduced.

Our conclusion is that the standardized method for map retrieval should be implemented in order to maintain an up-to-date technology in the management of GIS-data. The best way of using caching would be to use a proxy so that a user can take advantage of the data retrieved by other users. However this might cause problems regarding the ownership of the data. A way to solve this could be to make a business model with the map distributor.
Preface

This report is a result of our master thesis in Environmental Engineering at Luleå University of Technology. This thesis is the last step in our five years of studies to achieve our Master of Science degree at the division of Geographic Information Technology. This work has been done in the spring of 2002 at the Center for Distance-spanning Technology (CDT) at Luleå University of Technology.

We would like to thank our examiner, Christian Lundberg for assistance with technical support and helpful advice regarding this report. We would also like to thank Kåre Synnes and the other personnel involved during this master thesis and our opponents Patrik Wallgren and Johan Wikström. Finally our thanks go to Peter Berglund at Metria and Anders Haraldsson at Cartesia for taking their time with telephone interviews.

Luleå 2002-06-14

_________________________  _______________________
Magnus Nilsson              Per Nilsson
# Table of contents

1. **INTRODUCTION** ........................................................................................................ 4  
   1.1 **BACKGROUND** ................................................................................................. 4  
   1.2 **OBJECTIVE AND GOAL** ................................................................................... 5  
   1.3 **METHOD** ......................................................................................................... 5  
   1.4 **STRUCTURE OF THE REPORT** .......................................................................... 6  

2. **THEORY** .................................................................................................................. 7  
   2.1 **RETRIEVAL OF MAP DATA** .............................................................................. 7  
      2.1.1 Current method used in Alipes project ......................................................... 7  
      2.1.2 Standardization .............................................................................................. 9  
      2.1.3 Standardized WMS method .......................................................................... 10  
      2.1.4 Basic Service Elements within the WMS specification ............................... 11  
      2.1.5 Web Map Service Operations ...................................................................... 14  
   2.2 **CACHING** ......................................................................................................... 17  
      2.2.1 Description and benefits of caching ............................................................. 17  
      2.2.2 Caching methods and what to cache ......................................................... 18  
      2.2.3 Costs and rights for cached map data ....................................................... 20  
   2.3 **POSITIONING AND TRANSFERRING METHODS** ........................................... 21  

3. **IMPLEMENTATION** ............................................................................................... 24  
   3.1 **DELIMITATIONS** ............................................................................................. 24  
   3.2 **SYSTEM ARCHITECTURE** ............................................................................... 24  
   3.3 **CACHING** ....................................................................................................... 26  
      3.3.1 Caching on disc .............................................................................................. 26  
      3.3.2 Server caching ............................................................................................... 26  
      3.3.3 Implemented cache functions ...................................................................... 27  
   3.4 **WMS MAP CLIENT** ......................................................................................... 30  
      3.4.1 Technical platform ......................................................................................... 30  
      3.4.2 Developing method ....................................................................................... 31  
      3.4.3 Description of classes in application ......................................................... 32  

4. **DISCUSSION AND FUTURE WORK** .................................................................... 34  

5. **CONCLUSION** ....................................................................................................... 37  

6. **REFERENCES** ....................................................................................................... 38  

7. **APPENDIX** ............................................................................................................ 41  
   7.1 **HOW TO USE WMS MAP CLIENT** ................................................................. 41  
   7.2 **FILES** ............................................................................................................... 48  
   7.3 **JAVA DOCUMENTATION** ............................................................................. 49
1. Introduction

This master thesis consists of three main parts. The first part is to investigate and evaluate possibilities to transfer geographic information from different databases to a mobile user in a standardized way. Part two is to examine GIS information caching, to avoid high data costs. Finally the thesis deals with the design and implementation of an application that uses this standard and caching.

1.1 Background

This thesis is made for the Alipes project. Alipes is a project at the Center for Distance-spanning Technology (CDT). CDT is a department at Luleå University of technology. Their mission is to develop competence, products, and business opportunities at the intersection of computer and communications technology. CDT’s goal is to strengthen the competitiveness of companies and business in the region of northern Scandinavia.

Alipes project presents different services in mobile positioning, such as Location Aware Applications, which automatically trigger information that is relevant to the user's current location.

The positioning of the mobile devices is done by technologies such as WLAN (Wireless Local Area Networking), MPS (Mobile Positioning System), GPS (Global Positioning System) and Bluetooth. The Alipes architecture is divided into four sections, the Positioning Platform mentioned above, the Privacy and Security handler, the Map Service and the Service Infobase (Nord, et al., 2001). The architecture also supports peer-to-peer communication to allow clients to interchange position information over a local wireless network such as Bluetooth or WLAN.

Some of these Location Aware Applications also need a map for visualization. Today this map management is done through a simple but proprietary http API (Application Programming Interface) that retrieves map information from a database. However, there are occasions when you might need to retrieve geographic data from different servers. If these servers don't use the same queries for map retrieval problems will occur. Therefore a standardized method to gather this information is needed.

A problem with the currently net-based map retrieval is that it’s very expensive and therefore prohibitive for usage in Location Aware Applications. That is; there is a need to reduce the number of request to the map server. The use of caching map data, locally or proxy could be a way to reduce the cost that relates to these requests. Another issue when caching can be useful is when there is a limitation in the bandwidth, and you are handling large amount of data. This is often the case when dealing with geographic data.
1.2 Objective and goal

This master thesis focuses on mainly two areas that are related to the Alipes project, namely transferring of map data and caching.

Objective:

- To investigate a suitable method/standard for transferring GIS data to mobile users.
- Investigate whether it’s suitable to take map data directly from the distributor or if it’s possible to use caching for reducing the number of accesses to the map server.
  - Investigate if cached information should be stored on the server side or the client side (proxy or local caching).

Goal:

- To introduce a new map retrieval system for Location Aware Applications for the Alipes project.
- To find a solution regarding how to cache GIS data.
- To demonstrate these areas by building a simple application.

1.3 Method

The work is divided into two main parts, investigation and implementation. Initially information is gathered about standards and caching as a support for investigation and analysis. This information is retrieved via Internet and publications.

From the acquired information a summarization is made in order to find a standard for transferring geographic information to mobile users. The standard is evaluated and a comparison between the found standard and the currently used system in Alipes is made. A conclusion of possible benefits with using the standardized method is then made.

Caching methods found in the literature study are compared. The possibilities for reusing map data that have been cached and how some map distributors see this use is investigated. Further, the need of quality vs. cost for data is studied (i.e. it might be reasonable to only show every other pixel in the map when zooming). After an evaluation of the different methods a decision is made about which method is most suitable, local or proxy caching.
1.4 Structure of the report

This report is basically divided into five different chapters. The first, this one, presents a background and an introduction to the Alipes project and describe the purpose of this thesis. The second describes the theory concerning map requests, how it’s currently done within the Alipes platform and how the standardized Web Map Service (WMS) requests are done; it also contains a summary of the most essential parts of the OGC Web Map Service Implementation Specification (OGC, 2002a). Further, this chapter describes a basic description of what caching is and how it can be done. The third chapter describes the process of implementing an application that uses standardized map requests and caching. Finally, the last two chapters describe our conclusion of the benefits and possible drawbacks of using WMS and caching, and also describe how an application can use WMS and what enhancements can be done to our application.
2. Theory

2.1 Retrieval of Map Data

There are several ways of connecting to a map server, which means that an application developer must have knowledge on how to make requests for each different server. A standardized method may provide the developer an easier way to connect to several different servers. This chapter describes how the map requests are currently done within the Alipes project and further a standardized way of doing this.

2.1.1 Current method used in Alipes project

In Alipes a hypertext transfer protocol (HTTP) interface is used to retrieve maps for a specified position. The position is expressed in different geodetic data planes and coordinate systems. The queries to the map database are done by a simple URL request that include parameters such as map size, image type, scale and orientation. The map database is currently located at Luleå University of Technology (LTU). The map database is implemented to return the best available map, depending of the scale of the requested map (Nord et al., 2001). A description of the request and flow of events is given below.

- **Detailed HTTP request description**
  Microsoft Internet Information Server is used as Web-server. TCP/IP does the communication between the map-application and the server.
  The Web-server addition, esriMap.dll, is installed under the root- catalogue in the Web-server.
  EsriMap.ini specifies the IP- address, port etc. of the server, see table 2.1 (Chang, 1997).

<table>
<thead>
<tr>
<th>&quot;esrimap.ini&quot;</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GIS08]</td>
<td>Name of the application. Specified in the URL</td>
</tr>
<tr>
<td>server=130.240.5.183, 5062</td>
<td>IP-address and port for TCP/IP-communication</td>
</tr>
<tr>
<td>Timeout=20</td>
<td>Number of seconds the request remains to the server</td>
</tr>
<tr>
<td>retry</td>
<td>Number of times to reconnect to the server.</td>
</tr>
<tr>
<td>maxpending</td>
<td>Number of request that the server can handle, &quot;Server too busy&quot;</td>
</tr>
</tbody>
</table>
The map is downloaded when a client makes a request to the server, described by an URL (table 2.2)

**Table 2.2 Description of URL**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Http://&lt;server&gt;/scripts/esrimap.dll?</td>
<td>A standard syntax for an extension of MS Internet Information server</td>
</tr>
<tr>
<td>name=Alipes</td>
<td>Specifies the application that will receive the request, name is connected to the name in the esrimap.ini file.</td>
</tr>
<tr>
<td>system=rt90-meters</td>
<td>Reference system</td>
</tr>
<tr>
<td>pos</td>
<td>Position of the center of map.</td>
</tr>
<tr>
<td>size</td>
<td>The size of the map to be shown</td>
</tr>
<tr>
<td>scale</td>
<td>Which scale factor to use.</td>
</tr>
<tr>
<td>orientation</td>
<td>Rotation of the map (in degrees).</td>
</tr>
<tr>
<td>format</td>
<td>Type of image output.</td>
</tr>
<tr>
<td>symbolposition</td>
<td>A marker at a specified position</td>
</tr>
<tr>
<td>&amp;</td>
<td>Arguments are separated with &quot;&amp;&quot;.</td>
</tr>
</tbody>
</table>

An example of an http request to query a database for map information:
http://gitpc2.sb.luth.se/scripts/esrimap.dll?name=Alipes&system=rt90-meters&pos=7291226,1792593&size=500,500&scale=30&orientation=0&format=jpeg&symbolposition=7291956,1792893

- **Flow of events**
  1. A client makes a request to the server.
  2. The web server reads the esrimap.ini file in order to decide which map server to communicate with, and the connection is established.
  3. The map application creates a map based on the arguments sent with the query.
  4. The map server sends back the map to the web server.
  5. The web server sends the map to the client.
  6. The connection is ended.

*Figure 1. Flow of events when retrieving a map*
2.1.2 Standardization

There is a need for geographic information standards for all businesses that produces, distributes or utilizes spatial and non-spatial information. Therefore the work for developing these standards is important (ISO, 2001a). The International Organization for Standardization (ISO) is a worldwide non-governmental organization, whose work results in international agreements, which are published as international standards (ISO, 2001b).

Since digital data from a wide variety of sources is used by a diversity of applications a standard for geographic information and a service to proceed with this information has been developed. There are several advantages by using such a standard for the map management in applications. For instance, it will increase the understanding and use of geographic information and also increase the access and sharing of geographic data. The availability for collecting map data from different map servers can be increased when the request is defined to be of the same syntax for all map servers (ISO, 2001a).

For the geographic area the standards are based on specifications developed by The OpenGIS consortium (OGC). OGC is an industry consortium aimed at growing interoperability for technologies involving spatial information and location. It consists of more than 220 companies, government agencies and universities. Protocols defined by OpenGIS Specifications support solutions such as wireless and location-based services (OGC, 2002b).

OGC has produced a standardized method for map retrieval called Web Map Service Implementation Specification. The standard is applicable to produce maps of graphical formats, but not actual feature data or coverage data values.
2.1.3 Standardized WMS method

The OGC Web Services (OWS) suite includes three principal types of georeferenced information access service: Web Map Service (WMS), Web Coverage Service (WCS), and Web Feature Service (WFS).

A Web Map Service (WMS) produces maps of georeferenced data. These maps are generally rendered in a pictorial format such as Graphics interchange Formats (GIF), Portable Network Graphics (PNG) or Joint Photographics Expert Group (JPEG).

The WMS specification defines three operations: GetCapabilities, GetMap and GetFeatureInfo (see chapter 2.1.5). A map is said to be "queryable", if a WMS allow the GetFeatureInfo operation. This means that there is additional information about the objects on the map.

It also defines a syntax for World Wide Web (WWW) Uniform Resource Locators (URLs) that handles these operations. The content of such URLs depends on what task is requested. All URLs contains of a version number and the request type parameter. Depending on which of the three operations chosen, different additional parameters will be added to the URL (see chapter 2.1.4).

Using this standard will allow individual map layers to be requested from different servers. The WMS specification thus enables the creation of a network of distributed map servers. Many web mapping sites gathers all of its data in one place and makes it accessible only by their own developed interface. A WMS provider in a distributed WMS network, on the other hand, only need to manage it's own data collection which can be used by any WMS client.
2.1.4 Basic Service Elements within the WMS specification

The basic requests are the same for the three operations: GetMap, GetCapabilities and GetFeatureInfo. The basic service elements are described in the following text, which is a summarization from the WMS specification (OGC, 2002a).

- **Version Number and Negotiation**
  In order for the server and client to communicate there has to be a way to know that they understand each other. The specification demands that the version number that the client requests is equal to the one on the server, otherwise a version number negotiation will occur. The process is repeated until a mutual understood version is reached. If no mutual version is found negation has failed and the communication with the server is closed.

- **General HTTP Request Rules**
  HTTP supports two request methods: GET and POST. The basic WMS specification only defines GET for invoking operations. There are a number of characters that the WMS specification explicitly reserves for use in the query portion of the HTTP GET requests. When these characters appear in one of the roles defined in table 2.3, they are to appear literally in the URL.

<table>
<thead>
<tr>
<th>Character</th>
<th>Reserved usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Separator indicating start of query string</td>
</tr>
<tr>
<td>&amp;</td>
<td>Separator between parameters in query string</td>
</tr>
<tr>
<td>=</td>
<td>Separator between name and value of parameter</td>
</tr>
<tr>
<td>/</td>
<td>Separator between MIME type and subtype in format parameter value</td>
</tr>
<tr>
<td>:</td>
<td>Separator between Namespace and Identifier in SRS parameter value</td>
</tr>
<tr>
<td>,</td>
<td>Separator between individual values in list-oriented parameters.</td>
</tr>
</tbody>
</table>
The URL shall be valid according to the HTTP Common Gateway Interface standard (CGI). A URL prefix is defined as a string including the protocol, hostname, port number (optional), path, a question mark (“?”) and one or more server specific parameters (optional) ending with an “&”.

The URL's components are summarized in table 2.4.

<table>
<thead>
<tr>
<th>URL Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Http://host[:port]/path?{name=value}&amp;</code></td>
<td>URL prefix of service operation. [] denotes 0 or 1 occurrence of an optional part: {} denotes 0 or more occurrences. The prefix is entirely at the discretion of the service provider.</td>
</tr>
<tr>
<td>name=value&amp;</td>
<td>One or more standard request parameter name/value pairs defined by an OGC Web Service (OWS). The actual list of required and optional parameters is mandated for each operation by the appropriate OWS specification.</td>
</tr>
</tbody>
</table>

- **Request Parameter Rules**
  Parameter values are case sensitive, but parameter names are not. The parameters in a request don’t have to come in a specified order.
  To separate items in a parameter list, the comma (“,”) should be used. Individual entries in a parameter list may be empty, represented by two successive commas (“,”).

- **Common Request Parameters**
  There are several different parameters for a common request; the basic requests are described more detailed below. Additional parameters, Time, Elevation, Other Sample Dimensions and Vendor Specific parameters are not as frequently used and therefore not described further in this thesis.

  - **Version**
    Specifies the protocol version number, for example: &VERSION=1.1.0.

  - **Request**
    Indicates which service operation is being invoked, for example: &REQUEST=GetMap.
Theory

- **Format**
  Specifies the output format of the response to an operation, for example &FORMAT=image/gif. The WMS advertise in its Capabilities XML (eXtensible Markup Language) those formats it does support and shall accept request for any format it advertises.

- **Spatial Reference System**
  A Spatial Reference System (SRS) names a horizontal coordinate reference system code. There are two namespaces, EPSG and AUTO:
  EPSG stands for The European Petroleum Survey Group, which has compiled and distributed a set of parameters defining various geodetic and cartographic coordinate systems to encourage standardization across the Exploration and Production segment of the oil industry.
  An SRS request could look like this: 
  
  
  
  
  The AUTO namespace is used for a class of projections that include an arbitrary center of projection. Here the client specifies where the origin of the coordinate system shall be.
  For example, a request for a map centered at 100 degrees West longitude and 45 degrees North latitude will look like: 

- **Bounding Box**
  Bounding Box (BBOX) specifies a rectangular area, of which part of the world to be shown. The values in the BBOX parameter specify the minimum x, minimum y, maximum x, maximum y in that order (Figure 2). Some coordinate systems, for example RT 90 in Sweden, may use other axes order than X=East, Y=North.
  A query for such a boundary box may look like: 

  
  
  
  
  Figure 2. Bounding Box
2.1.5 Web Map Service Operations

How the WMS operations are implemented in the HTTP Protocol is described in the following section, which is a summarization from the WMS specification (OGC, 2002a).

- **GetCapabilities**
  GetCapabilities is a required operation that returns service-level metadata, which is a description of the WMS's information about the service itself and specific information about the available maps. It also gives information about the acceptable request parameters. Table 2.5 shows parameters for the GetCapabilities operation.

  Table 2.5 Parameter of a GetCapabilities request URL

<table>
<thead>
<tr>
<th>Request Parameter</th>
<th>Description</th>
<th>Required/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERSION</td>
<td>Requests version</td>
<td>Optional</td>
</tr>
<tr>
<td>SERVICE</td>
<td>Service Type</td>
<td>Required</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Request name</td>
<td>Required</td>
</tr>
<tr>
<td>UPDATESEQUENCE</td>
<td>Sequence number or string for cache control</td>
<td>Optional</td>
</tr>
</tbody>
</table>

A complete GetCapabilities request could look like this:
http://gitpc2.sb.luth.se/scripts/esrimap.dll?name=WMS&VERSION=1.1.0&SERVICE=WMS&REQUEST=GetCapabilities

... - <WMT_MS_Capabilities version="1.1.0" updateSequence="0">
  + <Service>
  - <Capability>
    - <Request>
      + <GetCapabilities>
      - <GetMap>
        <Format>image/gif</Format>
        <Format>image/jpeg</Format>
        <Format>image/png</Format>
        <Format>image/wbmp</Format>
      + <DCPType>
      </GetMap>
    + <GetFeatureInfo>
  + <Service>
  </WMT_MS_Capabilities>

*Figure 3. XML answer from GetCapabilities request*
• **GetMap**

GetMap returns a map image or a set of graphical elements. By HTTP GET operation the URL is invoked on the WMS.

The map request is described in table 2.6.

<table>
<thead>
<tr>
<th>Request Parameter</th>
<th>Description</th>
<th>Required/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERSION</td>
<td>Requests version</td>
<td>Required</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Request name</td>
<td>Required</td>
</tr>
<tr>
<td>LAYERS</td>
<td>Comma-separated list of one or more map layers</td>
<td>Required</td>
</tr>
<tr>
<td>STYLES</td>
<td>Comma-separated list of one rendering style per request layer</td>
<td>Required</td>
</tr>
<tr>
<td>SRS</td>
<td>Spatial Reference System</td>
<td>Required</td>
</tr>
<tr>
<td>BBOX=minx, miny, maxx, maxy</td>
<td>Bounding box corners in SRS units</td>
<td>Required</td>
</tr>
<tr>
<td>WIDTH</td>
<td>Width in pixels of map picture</td>
<td>Required</td>
</tr>
<tr>
<td>HEIGHT</td>
<td>Height in pixels of map picture</td>
<td>Required</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Output format of map</td>
<td>Required</td>
</tr>
<tr>
<td>TRANSPARENT</td>
<td>Background transparency of map</td>
<td>Optional</td>
</tr>
<tr>
<td>BGCOLOR</td>
<td>Hexadecimal red-green-blue color value for the background color</td>
<td>Optional</td>
</tr>
<tr>
<td>EXCEPTIONS</td>
<td>The format in which exceptions are to be reported by the WMS</td>
<td>Optional</td>
</tr>
<tr>
<td>TIME</td>
<td>Time value of layer desired</td>
<td>Optional</td>
</tr>
<tr>
<td>ELEVATION</td>
<td>Elevation of layer desired</td>
<td>Optional</td>
</tr>
<tr>
<td>Other sample dimension(s)</td>
<td>Value of other dimensions as appropriate</td>
<td>Optional</td>
</tr>
<tr>
<td>Vendor-specific parameters</td>
<td>Optional experimental parameters</td>
<td>Optional</td>
</tr>
</tbody>
</table>

A complete GetMap request could look like this:

```
```
• **GetFeatureInfo**
The GetFeatureInfo operation makes it possible for a client to choose a point on the map to obtain more information. It provides the ability for a user to specify which pixel is being asked about, which layer should be investigated, and what format the information should be returned in. It is only supported for the layers whose attribute "queryable" is set to true ("1").
Table 2.7 describes the parameters of GetFeatureInfo.

<table>
<thead>
<tr>
<th>Request Parameter</th>
<th>Description</th>
<th>Required/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERSION</td>
<td>Requests version</td>
<td>Required</td>
</tr>
<tr>
<td>REQUEST</td>
<td>Request name</td>
<td>Required</td>
</tr>
<tr>
<td>&lt;map_request_copy&gt;</td>
<td>Partial copy of the Map request parameters that generated the map for which information is desired.</td>
<td>Required</td>
</tr>
<tr>
<td>QUERY_LAYERS</td>
<td>Comma-separated list of one or more layers to query.</td>
<td>Required</td>
</tr>
<tr>
<td>INFO_FORMAT</td>
<td>Return format of feature information (MIME type)</td>
<td>Optional</td>
</tr>
<tr>
<td>FEATURE_COUNT</td>
<td>Number of features about which to return information</td>
<td>Optional</td>
</tr>
<tr>
<td>X</td>
<td>X coordinate in pixels from upper left corner</td>
<td>Required</td>
</tr>
<tr>
<td>Y</td>
<td>Y coordinate in pixels from upper left corner</td>
<td>Required</td>
</tr>
<tr>
<td>EXCEPTIONS</td>
<td>The format in which exceptions are to be reported by the WMS</td>
<td>Optional</td>
</tr>
<tr>
<td>Vendor-specific parameters</td>
<td>Optional experimental parameters</td>
<td>Optional</td>
</tr>
</tbody>
</table>

A complete GetFeatureInfo request could look like this:

```plaintext
```
2.2 Caching

Because of the high cost of net-based map retrieval there is use for a method to store the map images for further use in Location Aware Applications. That is, the map image needs to be stored so that more users than the one who fetched it can make use of it, and that it may be possible to reuse the downloaded map image.

There are different ways of storing (caching) data. The most common is to cache it locally (on disc) or to use a proxy server where other users can access the data.

2.2.1 Description and benefits of caching

In the WWW, resources are accessed via protocols (mini-language that computer program uses to talk to each other over the network), such as Hyper Text Transfer Protocol (HTTP) and the File Transfer Protocol (FTP). Which protocols that are used can be seen in a Uniform Resources Locator (URL) for example: http://www.luth.se indicates that HTTP protocol is to be used.

The browser opens a connection to the remote server to receive the resource. The basic idea behind caching is to keep copies of the things, which have recently been looked at (Ariadne, 1996).

A Web Cache sits between Web Servers and one or more clients, and watches for request for HTML (Hyper Text Markup Language) objects such as images, files etc. The cache saves a copy of the objects for itself. Then if there is another request for the same object, it will use the cached copy instead of asking the origin server again.

There are two main reasons for using web caching:

- Reduce the latency
  the time for the client to get the object is less then it would be asking the origin server, this makes the Web sites seem more responsive.

- Reduce the traffic
  because the object is only gotten from the server once, the bandwidth used by a client will be reduced. Reduced bandwidth can save money if the clients are paying by traffic, and it can also keep their bandwidth requirements lower and more manageable (Nottingham, 2002).
2.2.2 Caching methods and what to cache

A description of different caching possibilities, what objects to cache and how to share information with others is done in this chapter.

- **Browser Caches**  
  Any modern browser (Internet Explorer, Netscape etc.) lets you use a section of your computers hard disk to store objects that you just seen. This can be very useful when a client hits the "Back" button to see a page he just seen, the information will be served instantaneously from the browser cache.  
  Browser cache works in a simple way; it will check to make sure that the information is up-to-date, usually once a session (Nottingham, 2002).  
  These cache settings, the cache size and how often to compare the cached page to the page on the network etc., can be set under the preference settings in your Browser.

- **Proxy Caches**  
  A proxy is a special HTTP server that runs on a firewall machine. The primary use of a proxy is to allow access to the Internet if you are within a firewall. The proxy waits for a request from a client from inside the firewall, then the proxy forwards the request to the remote server outside the firewall. Responses from the server are read by the proxy and then sent back to the client. The ability to cache documents also makes proxies attractive to those not inside a firewall (W3C, 1994).  
  Proxy caches works almost in the same way as browser cache, only in a much greater scale, often hundred or thousand users. Large companies and corporation often set them up on their firewall (Nottingham, 2002).  
  When an object is requested once, it can be served to a large number of clients. Once a client has requested an object from a server it will be stored locally on the proxies disc. When one of the clients sharing the proxy generates a request, the proxy searches its local storage for the requested objects. If the object is available locally it's sent to the client, if not the request is passed on to the remote server. If the proxy cache belongs to a hierarchy of caches it is first sent to another proxy server. Using proxy caching will improve the overall performance of the system as long as the hit ratio is sufficiently high (Ortega et al., 1997).
• **Peer-to-peer**
Peer-to-peer is simply put, sharing of computer resources and services by direct exchange systems. These resources and services include the exchange of cache storage, information, processing cycles, and disk storage for files. In a peer-to-peer architecture, computers can act as both clients and servers, assuming whatever role is most efficient for the network. This reduces the load on servers and allows them to perform specialized services (such as mail-list generation, billing, etc.) more effectively (peer-to-peer working group, 2002).

• **What objects to cache**
Dynamically generated pages doesn’t not benefit as much of caching as static pages. When a dynamic page is retrieved from a server and stored in a cache, and when the client reuses it, the information might not be the one that the client asked for (terena, 2001).
An example of a dynamic object is Common Gateway Interface (CGI), these objects can never be cached. CGI programs are used to query databases, this means that the objects may differ every time (Ariadne, 1996).
Another is Active Server Pages (ASP), which looks like ordinary HTML pages but is dynamically created on the server.
Real time data such as stock indexes, video distribution etc. that may change in time, should not be cached due to its nature.
Password protected pages that are protected with HTTP authentication will not be cached (Nottingham, 2002).
Static HTML pages that don’t change very often are useful to cache. However it is important to check if the cache is up-to-date with the page content.
For graphics, such as images etc., the file size are relative large and thus uses a lot of bandwidth and they rarely changes (e.g. static). Caching of these objects is therefore especially advantageous.
2.2.3 Costs and rights for cached map data

There are different ways for map distributors to charge users that request map data. One way is for the client to pay for each data access, which is every time the client passes a request to the server. Another way for the distributor is to charge a company or a corporation on a time basis (for example; a month), this group then gets a restricted number of users that can access and use the data at a fixed price (Haraldsson, 2002).

Caching map data on a proxy and let other clients reuse the data is something that some distributors sees as violating the law because of the copyright. Making a business model together with the map distributor could solve this. This model specifies number of users, what data to access etc. The traffic on the caching server is then logged so that the distributor sees who makes the request and how often. This also enables the distributor to verify that the map data is not distributed without authorization. By using this method the client get the benefits of caching by reducing the latency and the traffic. In this way the copyright isn’t violated (Berglund, 2002).

There are several difficulties in controlling the use of data when it is stored locally on disc, in a similar way as using a Browser cache. One problem is to see for how long the data lives, another is how many times the client uses it. Therefore these cached maps are difficult to keep under the control of the copyright (Haraldsson, 2002).
2.3 Positioning and transferring methods

- **GPS**
  There are several different methods to determine the position that one currently have. One of the most commonly known is the GPS (Global Positioning System). The GPS is a system that determines the position from satellites that was introduced in 1973 and has gone from a restricted military use to a wide public use.
  In 2000 the US Army removed the selective availability for the GPS signals, which was an interference of the signal supposed to degrade the signal no more than 100 meters any given direction (100 meters radius for a total circle 200 meters across) 95% of the time. Since then the standard accuracy of the GPS information is about 22 meters, which can be improved to a few meters by using relative measurements (Differential GPS (DGPS) 0.5-2 m, Real Time Kinematic GPS (RTK) 1-3 cm) (Alipes, 2001a).
  There are however several alternative positioning system such as GLONASS (Globalnaya Navigatsionnaya Sputnikovaya Sistema) and GALILEO (the common European satellite positioning effort which will according to plan be operational in 2008), but they are not currently as developed as GPS.

- **WLAN**
  Another way for positioning is WLAN, which is a Local Area Network, based on semi-short range radio technologies. Within the Alipes project one of the most popular solutions in that area is deployed, the IEEE 802.11b WLAN technology (Alipes, 2001b).

  1997 the IEEE founded the standard 802.11 (WLAN). It operates as well as the follower 802.11b in the license free 2.4 GHz band. Hence every one can run any equipment at the frequency band as long as the emitted level doesn’t exceed a certain level, in Sweden 100 MW.
  1999 the 802.11b standard was founded, what differs this from the 802.11 is mainly the speed. 802.11 only manage 2 megabit per second (Mbits) and 802.11b can theoretically handle 11 Mbits. However the actual speed is seldom as high as the theoretical, it’s often decreased as much as 50 percent (Bodin, 2002).

  These facts have made short-ranged radio networks as perhaps the biggest contender against the UMTS mobile telephony networks. WLAN will yield higher bandwidths than any mobile telephony system, due to the fact that the cells are smaller. This enables the use of mobile applications that can consume higher amounts of bandwidth, at the expense of lesser mobility (Alipes, 2001b).
Calculating the position with WLAN

The theoretical distances to all base station that are within range are calculated. This is done by using the signal strengths, which together with each base station position creates a circle.

Every distance is analyzed to find the two intersections (see figure 4) between the two circles.

If two circles don’t intersect, the intersection between the circles and a line between the two origins is used (see Figure 5).
When every pair of distance has been analyzed, the mean value of all intersections is calculated which gives the position (see Figure 6).

![Figure 6. Mean value of intersections](image)

The calculating of the position as shown above is taken from the Alipes project (Byström et al, 2001).

- **Other**
  Telephony systems such as GSM (Global System for Mobile Communications) and UMTS (Universal Mobile Telecommunications System) can also be used for positioning and research is continuously being done in that area (Alipes, 2001a).

  Bluetooth is the primary competitor to WLAN. It works on the same frequency band, but the range is much shorter (about 10 meters), and the speed only about ten percent. Although there are implementations where the range has been extended up to 100 meter, which makes it interesting to use in the same areas as WLAN (Bodin, 2002).

- **Positioning in Alipes project**
  Within the Alipes positioning platform a wide spectrum of these positioning technologies are used. It includes GPS, DGPS, UMTS, IR, WLAN, HiperLan and Ericsson Mobile Positioning System (MPS) that uses GSM and GPRS (General Packet Radio Service).
3. Implementation

3.1 Delimitations

Due to the size and time limit of this thesis we have not implemented in our application any function that stores cached data on a proxy server. There can however be advantages in doing this, which is described in the report. We have instead chosen to cache locally on disk in order to reduce the number of requests to the map server. Also we have chosen to restrict the user to the same server for both map and additional info.

3.2 System architecture

This chapter describes the WMS map client briefly. Under the subsequent chapters a more detailed description is done.

- **WLAN**
  WLAN is used to position the client; it is also used to connect to the map server.

---

*Figure 7. System architecture*
• **WMS**  
  A Web Map Service (WMS) server is used to retrieve map images and additional information about objects on the map. This additional information can be restaurants, movie theatres etc. depending on which server is connected. When using WMS a standardized way of connecting to the server is used, this opens the possibility to connect to several different servers without changing the syntax for the requests.

• **Caching**  
  In this thesis caching locally on disk is done. This will reduce the number of requests to the server and therefore it will be more cost efficient than an ordinary map application that request a new map each time the user pan, zoom in etc. This cache function can be switched off if necessary.

• **WMS Map Client**  
  To interact with the program the client uses the Graphical User Interface (GUI) of the program (see Appendix 7.1 for details). The client retrieves its position from the WLAN - device, and due to the position; a map representing this area is downloaded from the WMS server. This map can then be used to query the WMS server for additional details or to orient the user.

The client and the caching method is the center of this work. These two parts have been investigated and developed in this thesis. The implementation of these areas is done in JAVA language.
3.3 Caching

In the following text different possible cache methods related to this project is described. The method used for caching implemented in the application is described more detailed in 3.3.3.

3.3.1 Caching on disc

Caching data locally on disc can be very cost efficient. Retrieving more information than being queried for and reusing it can make this possible. That is, if the application asks for a 100 by 100 pixel map and the cost of retrieval is the same for a 1000 by 1000 map then the bigger map should be fetched. A 100 by 100 map should then be cut out and sent to the application. If the user moves, the 1000 by 1000 map could be used again without fetching a new map from the server (or several map could be used to construct a bigger map). This reduces the queries to the map server and therefore reduces the costs.

3.3.2 Server caching

Using a server that sits between the application and the WMS can do caching of data on a server. This can be said to be a type of proxy caching where a user can take advantage of the data retrieved by other users. On the server a "library" of map images are created. The “books” in the library are map images that are stored with parts of its URL as filename. The calling application "looks" in the library for a map with a specific URL containing the requested BBOX. If one of the “books” fits the URL the map image is returned otherwise the query is sent to the WMS server.
3.3.3 Implemented cache functions

Caching is done locally on disc. The option “caching” enables the user to download a larger map than the one that fits the display (originalmap). This map can be called “cachedmap” (see figure 8) and is three times larger than the original. The “cache factor” (3 times) can easily be changed in order to suit the users. Pan, zoom in and zoom out functions can be seen in the following text.

- **Pan**
  When the client uses the pan function to center the map at a specific location, a new “pannedmap” is cut out from the “cachedmap”. A check is done to see if the pannedmap is inside the cachedmap, if not, a new map has to be fetched.

![Diagram of cachedmap](image)

*Figure 8. cachedmap*

![Diagram of Pan function](image)

*Figure 9. Pan function*
• **Zoom in**

In order to zoom in, a new “zoominmap” with a smaller BBOX, centered at the point where the user “clicked”, is created (see figure 10). This zoominmap is then rescaled to fit the size of the display (see Figure 11).

![Figure 10. Zoomin function](image1.png)

![Figure 11. Rescaled zoomin map](image2.png)
• **Zoom out**

To zoom out, a new “zoomoutmap” is created. The new map has a larger BBOX than the original map (see Figure 12) The zoomoutmap is then rescaled to the size of the display (see Figure 13).

*Figure 12. Zoomout function*

*Figure 13. Rescaled zoomout map*
3.4 WMS Map Client

A WMS map client application has been developed in this thesis. How to develop and how to implement the application in a PDA is described in the following text.

3.4.1 Technical platform

The market is full with different PDA’s (Personal Digital Assistants) or handheld devices, such as Palm, iPAQ, Casio, Handspring and more. The PDA can be used as a personal organizer or as an advanced computer with various functions such as word processors, browsers etc. It is also possible to expand most of these PDA’s with GPS, wireless connection to the Internet and more, depending on the trademark. In Alipes a Compaq iPAQ is used.

![Image](image1.png)

*Figure 14. The Gadget*

- **Hardware** (see figure 14)
  - Compaq iPAQ Pocket PC H3630
    - Memory: 32-MB SDRAM, 16-MB Flash ROM Memory
    - Processor: 206-MHz Intel StrongARM SA-1110 32-bit RISC Processor
    - Display: Color, 4096 colors (12bit) touch-sensitive liquid crystal display (LCD)
    - Operating System: Microsoft Windows CE (Pocket PC 2002 platform)
  - WLAN expansion pack
    - 11 Mbps
The application is developed using the Java Development Kit (JDK). The personalJava emulation environment helps to verify that the application will run on an implementation of the PersonalJava application environment (Sun, 2002a), which is used in the Alipes project.

To be able to implement the JAVA application to the IPAQ, Jeode EVM runtime engine from Insignia is used. EVM is tailored for the limited memory resources and browsing requirements of Pocket PCs, PDA’s and related smart handheld devices. EVM is fully compatible with the PersonalJava specification, and supports all PersonalJava 1.2 class libraries, including optional classes (Insignia, 2002).

- **Software**
  - Programming Language: JAVA (JDK 1.1.8)
  - Emulation environment: personalJava 3.0
  - Development environment: EMACS
  - Runtime environment in iPAQ: Jeode EVM

### 3.4.2 Developing method

The code in this thesis is developed using JDK on a PC; the source code is written in the developer environment called EMACS. The source code is then compiled and debugged using the Command Prompt on the PC. The JDK 1.1.8 can be downloaded for free from Sun’s homepage (Sun, 2002b), which is also the case with the personalJava 3.0.

In this application already developed classes from the Alipes project are used. These classes are for getting the position from the WLAN and for creating coordinates with longitude and latitude.

To include these classes a JAR file is used. A JAR file is a convenient way of packaging together a set of class files and any associated GIF images, data files, or other resource files. It consists of a normal ZIP format file, plus an optional manifest file describing its content (see Appendix 7.2).

The JAR file (Alipes_platform.jar) consists of the classes described above. Two other JAR files are also included, this to be able to handle XML (eXtensible Markup Language) and DOM (Document Object Model). The java_xml_pack-fall01 (containing xalan.jar and crimson.jar) can be downloaded free from Sun homepage(Sun, 2002c).

A JAR file (client.jar) is created containing the classes from the application. The JAR file is then stored under a catalogue in the PDA, together with the other JAR files needed for the program (Alipes_platform, xalan and crimson). In this catalogue a shortcut to the client.jar is created (see Appendix 7.2). This shortcut is used to set the path for the JAR files in the EVM environment and it’s also the startup icon for the program.

A simple click on the icon will start the application on the EVM environment.
3.4.3 Description of classes in application

There are mainly five classes in the application; UseMapper, Mapper, GetWMS, DrawMap and XmlSearch (see Figure 15). Further information about what these classes contain can be seen in appendix 7.3.

Figure 15. Model for classes
• **UseMapper**
The class for starting up and shutting down the application. It creates a new instance of the class Mapper, i.e. it calls on Mapper to create a new Graphical User Interface (GUI).

• **Mapper**
This class creates the GUI for the application (see Appendix 7.1). It handles events like button click and menu selections. When a connection to a server is established, Mapper requests what layers are available from the XmlSearch class. It then adds the available layers to the choice box on the “Map Settings” panel. The queryable layers will also be added to the choice box on the “Map” panel, so that the user easily can select which layer to query. For more info about the GUI, see Appendix 7.1.

• **DrawMap**
DrawMap is the class for creating the map. It uses the URL, created by class GetWMS, to connect to the WMS server in order to get an image of the requested map. This class handles map functions such as zoom in, pan etc. generated by user interaction with the map. Basically it creates a BBOX around the selected position, tells GetWMS to create a new URL, and then uses this to connect to the map server in order to get a new map. The cache function is also done here. If caching is selected a larger image is downloaded and used for cutting and rescaling as described in 3.3.3

• **GetWMS**
In this class the different URL’s are created. How these are built depends on which of the functions in WMS are selected. If the user asks the server for additional information about an object, a GetFeatureInfo request is generated to the WMS server. This class is also used to retrieve the position of the client.

• **XmlSearch**
Here the actual search in the XML document is done, using DOM. This class gets all the available layers from the WMS server; it also gets the layers that are query able (layers that have additional information).
4. Discussion and future work

- **Implementation issues**

When deciding which cache factor to use, that is how many times larger the cached map should be, one has to consider factors such as:

- what method to use to connect to the Internet (GPRS, WLAN etc.).
- size of memory in the device.

When using a WLAN device there is no problem regarding the download capacity, a large cached map can be downloaded without taking too long time, but using GSM will take too long in order to satisfy the user. And the longer it takes, the chance that the connection fails increase.

You also have to take into consideration if the cost for downloading from the Internet is expressed in number of access to the server or the number of bytes. If it is the latter then the use of downloading a larger map will have no cost benefits. The only gain then would be that the application seems more responsive.

In a handheld device such as iPAQ, the RAM is the factor that can decide the size of the cache factor. We tried to download a cached map three times larger than the original. This could be done without affecting the “speed” of the iPAQ if the scale was relatively small (city level), but as soon as the scale increased (country level), the size of the cached map became relatively large and sometimes cause memory problem in the PDA.

A problem during zooming in WGS84 has come to our knowledge. It appears that some servers do not have the accuracy needed to display the map in a fully satisfactory way, when the scale is down to 100’s of meters. This on the other hand isn’t a problem when using the Swedish coordinate system RT90.

Our theory is that there might be a correction error since WGS84 is a global system, and therefore may cause small calculation errors on country levels. Every country has a reference system of their own, to deal with this kind of problem.

The map destruction has to be considered when using the implemented “caching method”. If the quality of the map plays a decisive role this solution might not be recommended, but for the “every day user” this destruction is negligible (see Figure 25, Appendix 7.1).

Some layers will be affected by this caching method; if they are to be shown in a certain scale problem will appear. When zooming in there will not be request to the server for a new map, instead the “old” is cut out and rescaled. This means that the layers are shown according to the previous scale and information might be lost.

Another consideration is the relatively small display when developing an application to a PDA. Buttons, menus etc. must be carefully planned and organized to build a simple and easy to use interface.

Due to the screen size, the emphasis has first been put on making the map as large as possible, and secondly on the other layout.
Discussion and future work

- **Improvements on the application**
  The response from GetFeatureInfo can be difficult to handle, this because there might be problems displaying the response if a browser is not available (often the response comes as an HTML-page). In our case we had to remove all HTML-tags in order to display the response in a satisfactory way. The best thing would be for the server to respond with an XML-document, this way the developer can choose how to show the information in his own way, not controlled by a static page.

  Our WMS Map Client only support the JPEG format, this could however easily be changed. The response from the GetCapabilities states what format the server supports, and it would be desirable to let the client choose what format to use. This application only supports two reference systems, namely WGS84 and RT90. Modifications can however easily be made to adapt to other systems.

  A future work regarding the application can be to introduce several different servers, one for downloading the map and one for retrieving the additional parameters. There is also the possibility to use two or more servers for map retrieval, in this way an combination of layers from different maps can be made.

  We have in this thesis compared the methods of proxy caching and storing locally on the computer/PDA. Another way that we briefly mentioned for caching was peer-to-peer, where users can share information between each other in a network. As a future work this can be an area to look deeper into.

  - **Copyright issues**
    Much like the commonly known issue of music on mp3’s there is a blurred line on using caching of bought map data. This because caching is a relatively new and unknown area for the map distributors.

    One of the solutions for caching map data described in this report is to use a proxy for making the requests and saving the data there. Then a copy of this data is used when making a request for a previously requested map from a client application. This allows other users in the network to use data previously downloaded by someone else. However, this may cause conflicts with the copyright law and has to be solved with the map distributors. This can be done by making some kind of business model with the distributor that states who is allowed to use the map data and how often. Perhaps another solution could be a kind of time-to-live that is set in the cached map. The Swedish map distributors we talked to were not really aware of this possible solution of caching data and could not give any solution on how to solve it for the moment. The distributors also felt that this was some kind of copyright problem. However they have used some kind of business model where the users could access their map servers (not WMS) at an annual cost.
• Security
It is important to keep in mind that a request for a map or a position should not reveal the
users position so they might feel monitored or so that the information can be used for
unintended purposes.
In the Alipes platform a query to the platform for a position will automatically result in a
request to the application, and the application is then free to accept or deny the request or
to choose to ask the user if they wish to permit the query.
In the application developed in this thesis a request for the position need to be made for a
few occasions: First when the application is started, a map is centered around the users
position (this is also done with the option “Reload Map”, which refreshes the map) and
second when the option “tracking” is selected, where the users current location is
continuously shown on the map.
However, each time a request is made from the application to a WMS server (every time
a new map is downloaded or information about an object is received), one could guess
that the user is making requests for information about the area where he currently is. A
rough estimation of his whereabouts can then be done. This is a privacy issue that we feel
that the WMS specification should deal with.
5. Conclusion

- **Comparison of WMS and the current method in Alipes project**
  In the current method the user is not able to select map server, and is therefore restricted to use a map server that is not necessarily the most suitable for his wishes. Also if that server is temporarily out of use, he cannot receive a map without altering the source code.

As the numbers of distributors of geographic data using WMS are increasing, the Alipes-project will benefit increasingly from using this system. Geographic information, such as maps, addresses and other object data, will be easier to manage and there will be a possibility to change map distributor if needed.

Therefore, our conclusion is that the standardized method should be implemented in order to maintain an up-to-date technology in the management of GIS-data.

- **Caching**
  Since map data is static, it’s seldom needs updating and is therefore advantageous to cache. Hence, we feel that caching is useful to implement in the Alipes project.

  - **Selection of method for caching net-based map images**
    In the matter of caching data either locally or on a server, our choice would be to use the latter. This because of the capabilities for storing images is bigger on a server. The sizes of the image files are relatively large and may affect a handheld device negatively because of its limited resources (e.g. hard drive etc.). Another advantage is that different applications can use the same image from the server; this cannot be done if the image is stored locally.

  - **Caching in this thesis**
    We have found that the caching method implemented in this thesis is a way to reduce the number of requests to the map server and therefore reduce the costs for map retrieval. Further this method will make the application seem more responsive when there is no need to contact the server every time a map function is done.

  - **Map Destruction**
    In the zooming functions on the application destruction of the map is done when caching is selected. This destruction will increase with the number of times zooming is done. We have found that the destruction will be to great and affect the quality too much if more than one zoom in or zoom out is done from the same cached map. Because of this, a new cached map is downloaded if the client tries to zoom in or zoom out twice at the same map. There is on the other hand no problem if the user zoom in once, then zoom out to the original map and so on.
6. References

Alipes, 2001a, Location Aware Software in Networked Mobile Devices
http://www.cdt.luth.se/projects/alipes/alipes.html
Last visited: 2002-05-13

Alipes, 2001b, Advanced WaveLan Positioning
Last visited: 2002-05-13

Ariadne, 1996, Cashing in on Caching
Electronic magazine published by UKOLN
www.ariadne.ac.uk/issue4/caching/
Last visited: 2002-05-08

Berglund P, 2002, Phone interview about caching of map data etc.
Berglund Peter, Metria, SGD expert
2002-03-03

Bodin P, 2002
WLAN Stora Testet.
Nätverk och Kommunikation nr 8, 7 maj 2002

Byström A, Dahlroth T, Eriksson J, Fernandez L, Holmgren D, Johansson T, Larsson P,
Styf I, Ståhl M, Varillas N, Wu M, 2001
Advanced WaveLan Positioning System
Last visited: 2002-05-29

Chang K-P, 1997
The design of a web-based Geographical Information System for community participation.
http://www.owu.edu/~jbkrygie/krygie_html/lws/chang.htm
Last visited: 2002-05-30

GNRT, 2001, Guide to Network Resource Tools
Dynamic content and Caching
www.terena.nl/libr/gnrt/providing/caching.html
Last visited: 2002-05-08

Haraldsson A, 2002, Phone interview about caching of map data etc.
Haraldsson Anders, OnlineGIS Cartesia.
2002-02-20
Insignia, 2002, INSIGNIA JEODE PDA EDITION VM
http://www.insignia.com/content/products/pda.shtml
Last visited: 2002-05-24

Last visited: 2002-05-07

ISO, 2001b, International organization of Standardization
Last visited: 2002-05-07

Nord J, Synnes K, Parnes P, 2001
An Architecture for Location Aware Applications. Proceedings of the Hawaii
International Conference on System Sciences (HICSS-35), January 7 – 10, 2002, Big
Island, Hawaii, USA.
Last visited: 2002-05-24

Nottingham M, 2002
Caching Tutorial for Web Authors and webmasters
www.mnot.net/cache_docs
Last visited: 2002-05-08

OGC, 2002a, Open GIS Consortium
Web Map Service Implementation Specification (OGC 01-068r3), Version 1.1.1
http://www.opengis.org/techno/specs/01-068r3.pdf
Last visited: 2002-05-07

OGC, 2002b, Open GIS Consortium
http://www.opengis.org/index.htm
Last visited: 2002-05-07

Ortega A, Carignano F, Ayer S, Vetterli M.
June 23 --- 25, 1997, Princeton, New Jersey, USA
http://sipi.usc.edu/~ortega/SoftCaching/mmssp97.html
Last visited: 2002-05-08

peer-to-peer working group, 2002
what is peer-to-peer?
http://p2pwg.org/whatis/index.html
Last visited: 2002-05-24
Sun, 2002a, Sun microsystems
PersonalJava Emulation Environment
http://java.sun.com/products/personaljava/pj-emulation.html#pjee3.0.2
Last visited: 2002-05-07

Sun, 2002b, Sun microsystems
The source for Java technology
http://java.sun.com/
Last visited: 2002-05-24

Sun, 2002c, Sun microsystems
Java XML Pack
http://java.sun.com/xml/java/xmlpack.html
Last visited: 2002-05-24

W3C, 1994, World Wide Web Consortium
World-Wide Web Proxies
www.w3.org/History/1994/WWW/Proxies
Last visited: 2002-05-08
7. Appendix

7.1 How to use WMS Map Client

- **On Start-up**
  The first time the application is used, the user will be directed to the Server Settings menu in order to make the correct search path to a WMS server (see Server Settings below). Otherwise, the last settings before the application was closed (which will automatically be saved into a file "settings.ini"), is used to make a request to the WMS server for a map around the users current position.

*Figure 16. WMS Map Client*
• **Menus**
The “File” menu only contains an option “exit” (Figure 17), which will close the application after automatically saving the values in choices, lists and text fields.

![Figure 17. The menu “File”](image)

The menu “Menu” contains a list of available options for the user; Map View will show the map, Map Settings opens a panel with possibilities to select layers from the current WMS server (see Map Settings), Server Settings opens a panel where the path to the WMS server can be set, Set Position opens a panel where the user can enter a position of his own choice to center the map around. Finally Coordinate System lets the user see what coordinate system currently used.

![Figure 18. The menu “Menu”](image)
The menu “Options” gives the user some options for the map. When “Tracking” is selected “On” the user’s current position will be shown on the map until “Tracking” is deselected or the user uses map functions such as zoom and pan.

When “Caching” is selected “On” a larger map than the one shown is downloaded from the server. This will reduce the number of requests to the WMS server and affects the user in the way that “zoom” and “pan” functions will be quicker within this “cached” map, since no new map need to be downloaded until the requested area is outside the “cached” map.

The option “Reload Map” will refresh the map around the user’s current position.

The menu “Help” presents links to useful information about the application, such as a brief help about coordinate systems.

Figure 19. The menu “Options”
• **Server Settings**
This panel is used to create a path to the WMS server. The WMS Map Client is adjusted so that it applies to WMS version 1.0 and higher. For now, the application only supports the coordinate systems RT90 (SRS=EPSG:2400) and WGS84(SRS=EPSG:4326). Adjustments can however fairly easy be made so that it applies to other. Some servers use the same basic request for different types of maps (for instance ESRI has vegetation maps, land use maps and more only separated with the additional parameter &ServiceName= ”name of the service in question”). Therefore a field for adding additional parameters is available on the panel.

![Figure 20. Server Settings](image)

*Figure 20. Server Settings*
- **Map Settings**
  This panel gives the user the possibility to determine which layers that will be shown on
  the map. It is also possible to decide in which order the layers should be on the map.

  ![Figure 21. Map Settings](image1)

  **Figure 21. Map Settings**

- **Set Position**
  This panel is useful if the user wish to set an own position. This can be useful if the
  positioning should for some reason malfunction or be out of range, in which case no map
  is created on startup or on “Reload Map”.

  ![Figure 22. Set Position](image2)

  **Figure 22. Set Position**
• **Queryable layers**
Objects such as stores, restaurants etc can have additional information about opening hours, business info etc. By selecting the “query” function (the question mark “?”) and then clicking on the object, information can be seen about the object if so exists. A small demonstrative example is presented below.

![Queryable object “Centrumrestaurangen”](image)

_Figure 23. Queryable object “Centrumrestaurangen”_

By clicking on the object Centrumrestaurangen above, information about it is presented as seen in the figure below.

![Answer from WMS](image)

_Figure 24. Answer from WMS_
- **Caching**
  When using the method caching, there will be a destruction of the map quality. This is shown in Figure 25 and 26. You can see that the pixels in the destructed map are more apparent than in the original.

![Figure 25. Comparison of original and destructed map (zoomed in)](image)

![Figure 26. Comparison of original and destructed map (zoomed out)](image)
7.2 Files

A Jar file consists of the classes and a manifest file (see Figure 26). The manifest file describes the jar file; it states the name of the main class (UseMapper).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Modified</th>
<th>Size</th>
<th>Ratio</th>
<th>Packed</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manifest.inf</td>
<td>MF File</td>
<td>2002-05-07</td>
<td>1370</td>
<td>52%</td>
<td>663</td>
<td>meta-inf</td>
</tr>
<tr>
<td>XmlSearch.class</td>
<td>CLASS File</td>
<td>2002-04-05</td>
<td>3064</td>
<td>49%</td>
<td>1553</td>
<td></td>
</tr>
<tr>
<td>WindowListener....</td>
<td>CLASS File</td>
<td>2002-03-26</td>
<td>447</td>
<td>30%</td>
<td>312</td>
<td></td>
</tr>
<tr>
<td>UseMapper.class</td>
<td>CLASS File</td>
<td>2002-05-06</td>
<td>758</td>
<td>34%</td>
<td>498</td>
<td></td>
</tr>
<tr>
<td>SendPoint.class</td>
<td>CLASS File</td>
<td>2002-05-06</td>
<td>1381</td>
<td>40%</td>
<td>827</td>
<td></td>
</tr>
<tr>
<td>Mapper.class</td>
<td>CLASS File</td>
<td>2002-05-07</td>
<td>1632</td>
<td>52%</td>
<td>7813</td>
<td></td>
</tr>
<tr>
<td>Mapper$ButtonLi...</td>
<td>CLASS File</td>
<td>2002-05-06</td>
<td>4598</td>
<td>45%</td>
<td>2514</td>
<td></td>
</tr>
<tr>
<td>GetWMS.class</td>
<td>CLASS File</td>
<td>2002-05-06</td>
<td>7586</td>
<td>49%</td>
<td>3842</td>
<td></td>
</tr>
<tr>
<td>DrawMap.class</td>
<td>CLASS File</td>
<td>2002-05-07</td>
<td>13017</td>
<td>55%</td>
<td>5864</td>
<td></td>
</tr>
</tbody>
</table>

Figure 26. JAR file

A short cut to the main class (UseMapper) must be created in the Jeode environment, this in order to allow the user to start up the application by clicking on the shortcut (see figure 27).

Figure 27. A Shortcut file in Jeode VM

If another jar file is included it is added in the same way with name and path. The name of the main class is then put as the final statement in the file. The file is saved as a shortcut.
7.3 JAVA documentation

This is a draft from the javadoc created by java. It describes the content of the classes.

**Class UseMapper**

```java
public class UseMapper extends Object
```

**XmlSearch**

```java
public class XmlSearch extends Object
```

**XmlSearch**(Node)  
Constructor for class XmlSearch

**getLayers**()  
Method for getting all the layers from the XML document

**getQueryLayers**()  
Method for getting the queryable layers from the XML document
**Class Mapper**
java.lang.Object

+----java.awt.Component
+----java.awt.Container
    +----java.awt.Window
        +----java.awt.Frame
            +----Mapper

public class Mapper
extends Frame
implements ItemListener

**Mapper()**

action(Event, Object)
    Method for handling menu selections.
caching()
    Method to check whether caching is selected
coordSyst()
    Method to check which Coordinate System is being used
getAllQueryLayers()
    Method for getting all the queryable layers in the Choice "queryChoice"
getFunction()
    Method to check which function that is selected.
getLayerChoice()
    Method for getting all the layers in the Choice "layerChoice"
getLayerList()
    Method to get the layers selected in LayerList, used in url request.
getLayerListFormatted()
    Method to get the layers selected in the list "layerList"
ggetQueryLayer()
    Method that gets all the layers in the Choice "queryChoice", used for url request
getServerData()
    Method to get server data from textfields in settings.
isInLayerList(String)
    Method for checking if a Layer already exists in the list.
itemStateChanged(ItemEvent)
    Method for handling checkBoxMenuItem events
loadLabel()
    Method that loads a message showing "Loading Map"
loadSettings()
    Method for loading the last saved settings
saveSettings()
    Method for saving settings to file "settings.ini"
setCaching(boolean)
    Method that sets whether caching is enabled
setCoordSyst(String)
Method that sets which Coordinate System that is going to be used

```
setFunction(String)
```
Method that sets a String indicating which button that is pressed.

```
setInfo(String)
```
Method that sets what information that is to be shown when using the Feature_Info request

```
showError(String)
```
Method for displaying error messages when occurred

```
showInfo()
```
Method that shows the infoPanel from within other classes

```
unLoad_label()
```
Method that removes the message "Loading Map"

### Class GetWMS

```java
public class GetWMS
extends Mapper
```

```
createImgBBox()
```
Method for creating Image BBOX(cached) of the ordinary BBOX

```
firstTime()
```

```
getBBox()
```
Method for fetching the current BBOX

```
getCurrentPosition()
```
Method for getting the current position

```
getFeatureInfo(Point, String)
```
Method for creating the URL with the GetFeatureInfo request.

```
getImgBBox()
```
Method for fetching the current image BBOX(Cached)

```
getMap()
```
Method for creating the URL to the map server with the GetMap request on "start up" Is compatible for WMS server version 1.0.1.1.0 and future versions if request parameters are not changed

```
getMap(String)
```
Method for creating the URL with the GetMap request to the map server with different boundary boxes depending on mapActions Is compatible for WMS server version 1.0.1.1.0 and future versions if request parameters are not changed
Appendix

GetWMS()
getXMLDoc()
Method for creating the URL with the GetCapabilities request and saving the XML-document as "capabilities.xml" Is compatible for WMS server version 1.0,1.1.0 and future versions if request parameters are not changed

initializeBBox()
Method for initializing the BBOX

addListener()
Method for listening to position changes

setBBox(double, double, double, double)
Method to set the current BBOX

setPosition(Coordinate)
Method for setting the current position

setImageBBox(double, double, double, double)
Method to set the current Image BBOX(Cached)

Class DrawMap

java.lang.Object
| +----java.awt.Component
| | +-----java.awt.Canvas
| | | +----DrawMap

public class DrawMap
extends Canvas
implements MouseListener, MouseMotionListener, Runnable

DrawMap()
Constructor for creating a new instance of DrawMap Creates a map image depending on the url

destroy()
Method for stopping timer

drawAgain()
Method for drawing a new map image without creating a new instance of class DrawMap

getUrl()
Method to get the URL

mouseClicked(MouseEvent)
Method for handling mouseClicked Event

mouseDragged(MouseEvent)
Method for handling mouseDragged Event

mouseEntered(MouseEvent)
Method for handling mouseEntered Event

mouseExited(MouseEvent)
Method for handling mouseExited Event

mouseMoved(MouseEvent)
Method for handling mouseMoved Event

mousePressed(MouseEvent)
Method for handling mousePressed Event

mouseReleased(MouseEvent)
Method for handling mouseReleased Event
paint(Graphics)
run()
    Method for running timer, used to listen for position when tracking
setUrl(String)
    Method for setting the URL