MSIS Installation Guide

This document is intended as a guide to installing and configuring the MSIS application and its dependencies. The application contains server side components and different clients. This guide will go through the components step by step, starting with the server side components.

Contents

Server Side

The server side consists of three separate components: The database, web services and the MSIS User Administration Website. This guide will give the instructions to install and configure these components to run on a single server machine. First the requirements will be listed.

System Requirements

These are the requirements for running the server side components:

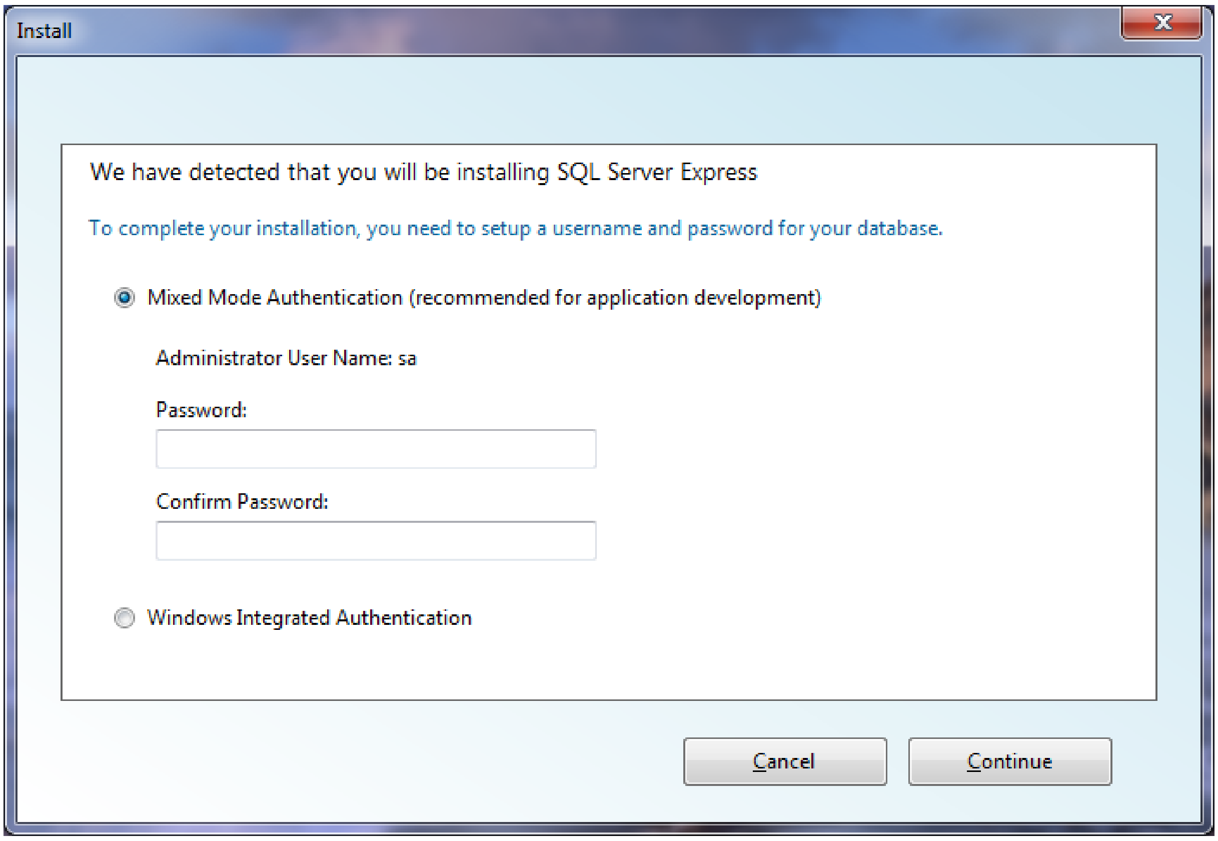
- Microsoft Windows Server 2003 or newer (Mainstream operating systems like Vista and 7 will also work, but Windows XP will not work).
- Microsoft IIS Web Server
- Microsoft SQL Server 2008 (or Express Edition), with Microsoft SQL Server Management Studio
- Microsoft .NET Framework 3.5

From the next section of the guide, it is required that Windows, IIS and .NET Framework are installed. However, the next section will go through the installation of SQL Server, so there is no demand that this is already installed at this time. IIS is available on the Windows installation media as an extra application (in the control panel by using "Turn Windows features on or off" in "Programs and Features". Remember to enable ASP.NET support as it is disabled by default). The .NET Framework is available to download from Microsoft's website (or already installed in Windows on computers running Vista or newer OS). To check if your version of .NET framework is updated, check the listing of programs in "Programs and Features" or check the listing of programs in the "Turn Windows features on or off". The framework version should be listed.

Database Setup

Microsoft SQL Server 2008 Express Edition is available for free from here: http://www.microsoft.com/express/sql/default.aspx. It is highly recommended to select the edition with Runtime and Management Tools (currently 230.4 MB).

During the installation it is recommended to select ***Mixed Mode Authentication*** as seen in Figure 1below. Make a note of your username and password, because it will be needed later in the database setup process.


Install

We have detected that you will be installing SQL Server Express

To complete your installation, you need to setup a username and password for your database.

[✓ Mixed Mode Authentication (recommended for application development)]

Administrator User Name: sa

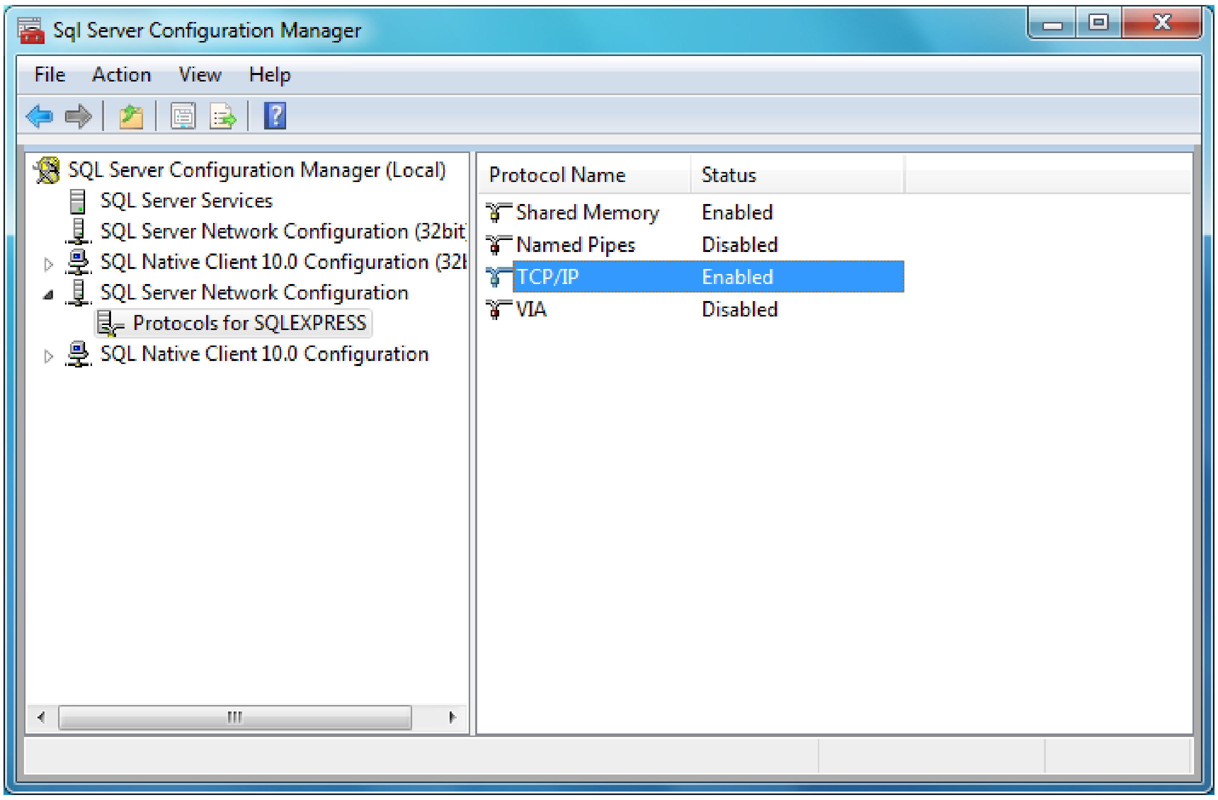
Password:

Confirm Password:

Windows Integrated Authentication

Figure 1: Select Mixed Mode Authentication

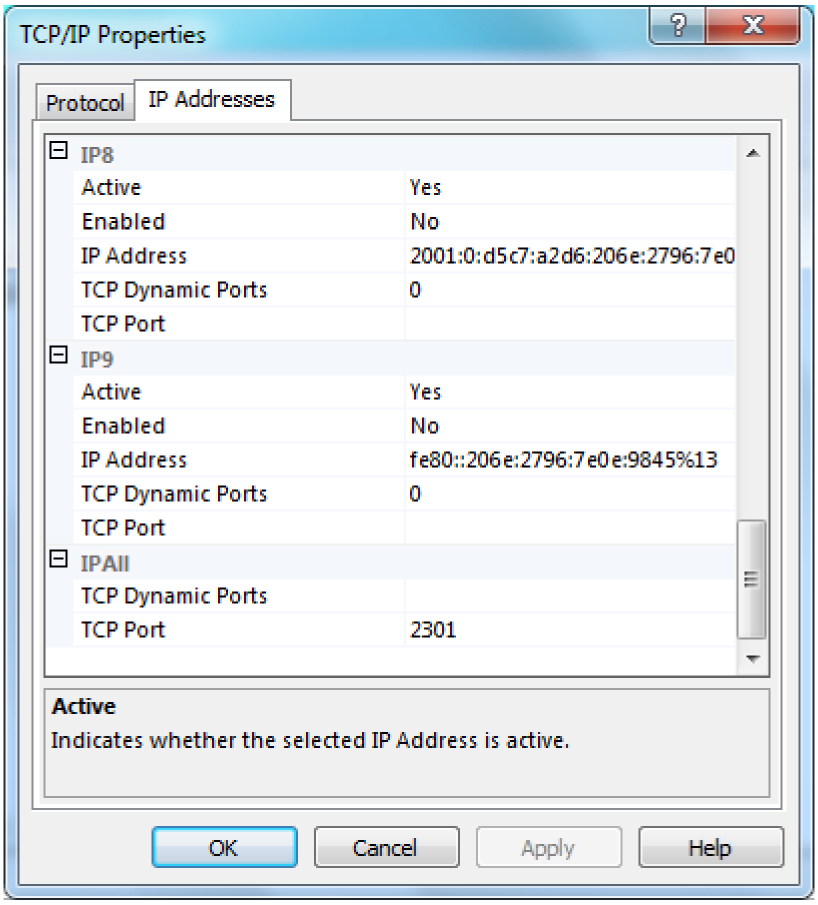
After the installation is completed, run ***SQL Server Configuration Manager*** from the SQL Server 2008 start menu group. In this application we need to enable TCP/IP browsing and set the TCP port number that we will use to connect to the database from the web services. There is a browser menu on the left pane of the application. Expand SQL Server Network Configuration and select ***Protocols for SQLEXPRESS***. Some elements will appear in the main panel. Select ***TCP/IP***, right click on it and select ***Enable***. When done, a result like in Figure 2 is expected.


Sql Server Configuration Manager
File
Action
View
Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit
SQL Native Client 10.0 Configuration (32l
SQL Server Network Configuration
Protocols for SQLEXPRESS
SQL Native Client 10.0 Configuration

Protocol Name	Status
Shared Memory	Enabled
Named Pipes	Disabled
TCP/IP	Enabled
VIA	Disabled

Figure 2: Enabling TCP/IP

After enabling TCP/IP, right click again and select ***Properties***. In the window that appears switch tab to IP Addresses and scroll all the way down to the bottom. There should be an element called IPAll with a sub-element called TCP Port. Set this value to 2301 as in Figure 3, or use a custom value if you wish to use other values than the guide.


TCP/IP Properties

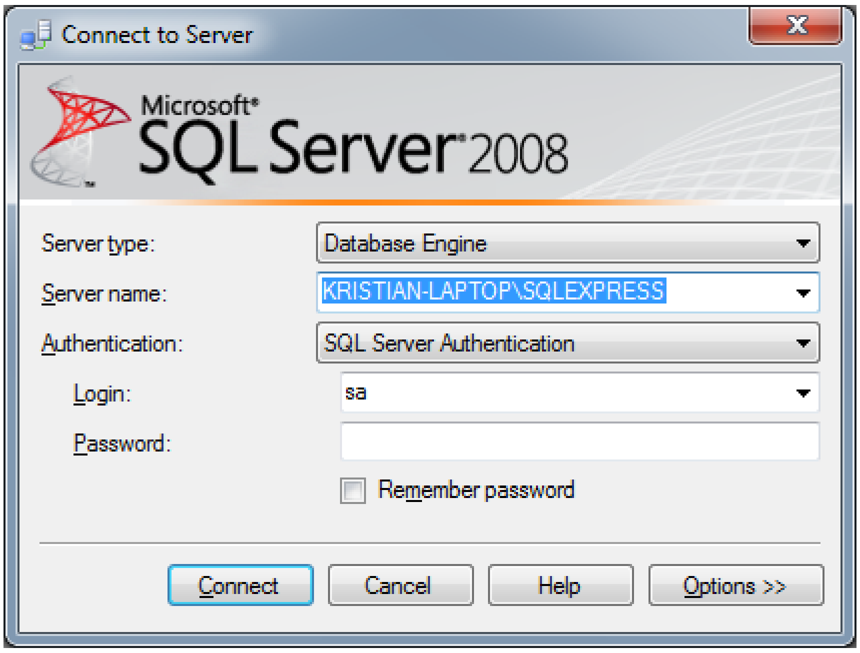
Protocol | IP Addresses

IP8	
Active	Yes
Enabled	No
IP Address	2001:0:d5c7:a2d6:206e:2796:7e0
TCP Dynamic Ports	0
TCP Port	
IP9	
Active	Yes
Enabled	No
IP Address	fe80::206e:2796:7e0e:9845%13
TCP Dynamic Ports	0
TCP Port	
IPAll	
TCP Dynamic Ports	
TCP Port	2301

Active
Indicates whether the selected IP Address is active.

Figure 3: Setting the TCP Port number

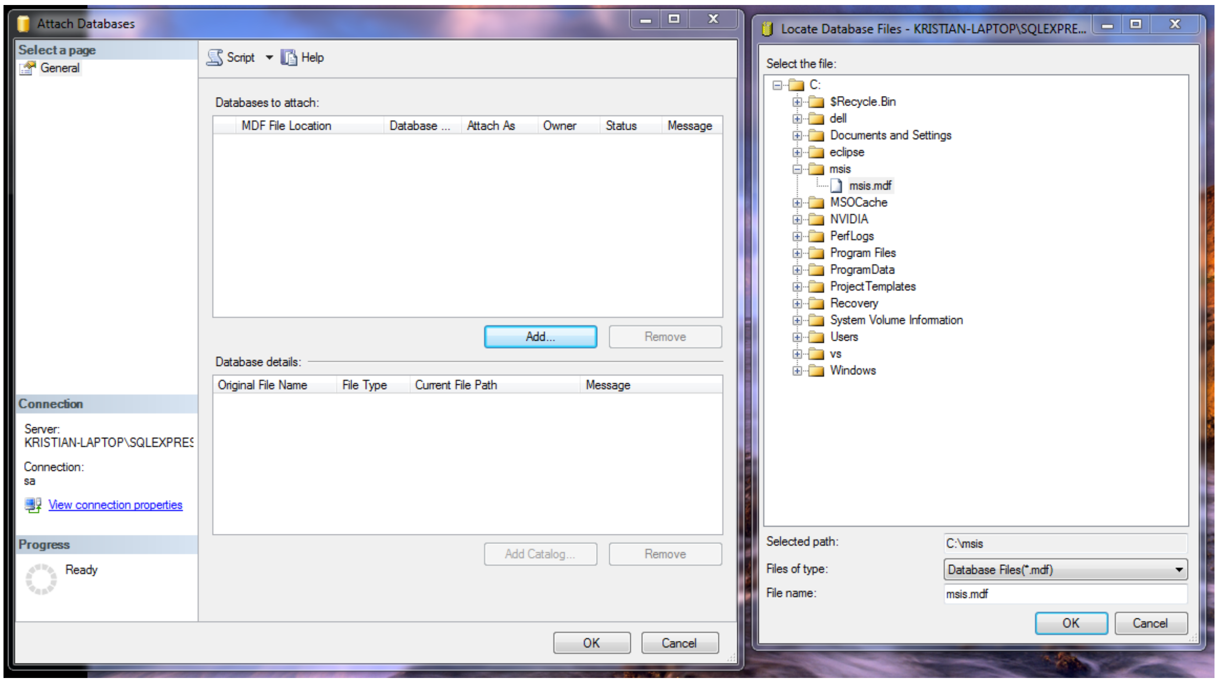
run Microsoft SQL Server Management Studio and log on using the username and password from the setup. Remember to set the Authentication selection ***to SQL Server Authentication*** as seen in Figure 4.



Server type: Database Engine
Server name: KRISTIAN-LAPTOP\SQLEXPRESS
Authentication: SQL Server Authentication
Login: sa
Password:
Remember password

Figure 4: Connecting to SQL Server using SQL Server Management Studio

When connected to the server, use the Object Explorer on the left side and right click on the element named Databases, and select ***Attach***. This will bring up a new dialog named Attach Databases. Then select ***Add***, which will open a file browser where you will have to locate the database file. This file is part of the delivered archive, and is by default named ***msis.mdf***.



Databases to attach:

MDF File Location	Database ...	Attach As	Owner	Status	Message

Database details:

Original File Name	File Type	Current File Path	Message

Server:
KRISTIAN-LAPTOP\SQLEXPRES
Connection:
sa

Select the file:

```
C:
  $Recycle.Bin
  dell
  Documents and Settings
  eclipse
  msis
    msis.mdf
  MSOCache
  NVIDIA
  PerfLogs
  Program Files
  ProgramData
  Project Templates
  Recovery
  System Volume Information
  Users
  vs
  Windows
```

Selected path: C:\msis
Files of type: Database Files(*.mdf)
File name: msis.mdf

Figure 5: Attaching a database file

After pressing ***OK***, the program will begin to import the database. Two common errors that may occur in this step are:

- The program should be running as Administrator if used on a system protected by User Account Control (Windows Vista or newer operating systems).
- The user doesn’t have proper permissions in the folder he tries to import from. Move file to a folder with more permission (e.g. public folder or another partition).

Afterwards the Object Explorer should be updated to reflect that a new database has been imported. Expanding the nodes should bring a list much like the one in Figure 6.

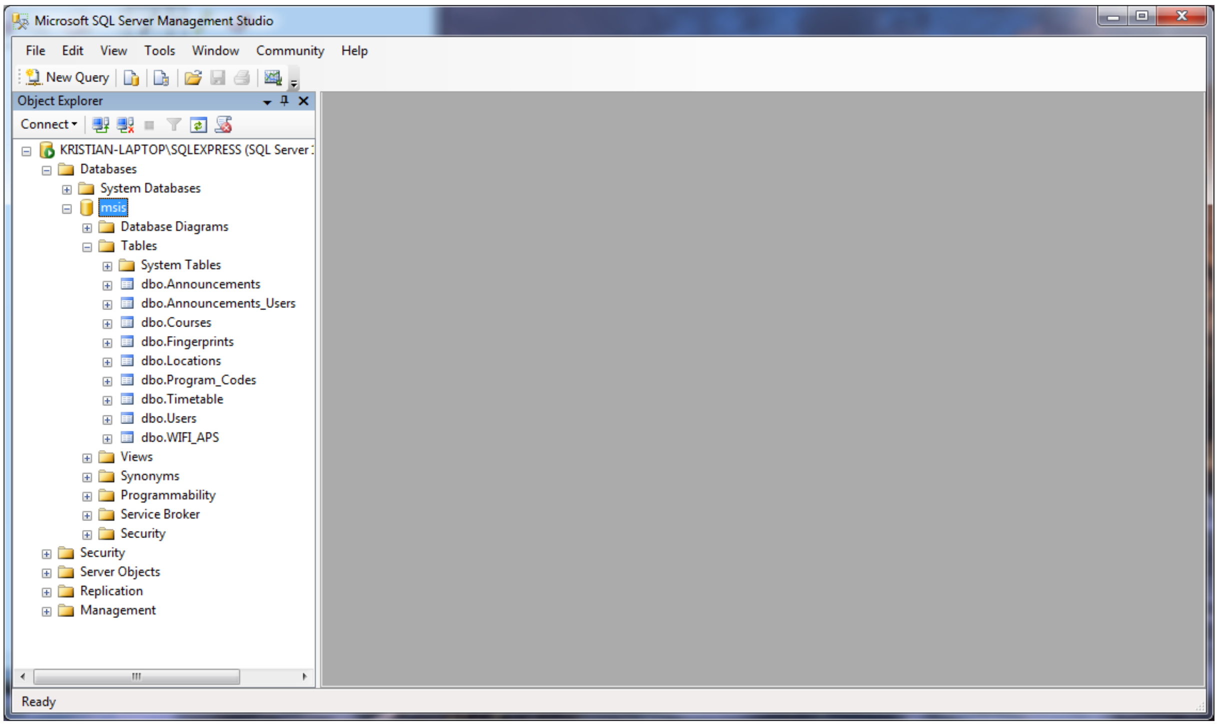


Figure 6: After the database has been imported.

That’s it for configuration of the database. Next up is the web services.

Web Services

The web services are located in the folder named ***Application_Server*** after unpacking the contents of the archive. Copy the contents of this folder to an appropriate place where you want to keep these files (e.g. make a folder ***msis*** under C:\, so that the path to the folder named webservices is: ***C:\msis\webservices***).

Afterwards, open ***Internet Information Services (IIS) Manager*** from Administrative Tools (or use search term IIS in start menu). In the left pane expand from the server name, expand ***Sites*** and expand ***Default Web Site***. Right click on Default Web Site and select ***Add Virtual Directory***. Name the virtual directory what you want and set the physical path to the folder where the web services are (remember to use the folder named webservices and not the parent directory). See Figure 7 for an example on how to add a virtual directory.

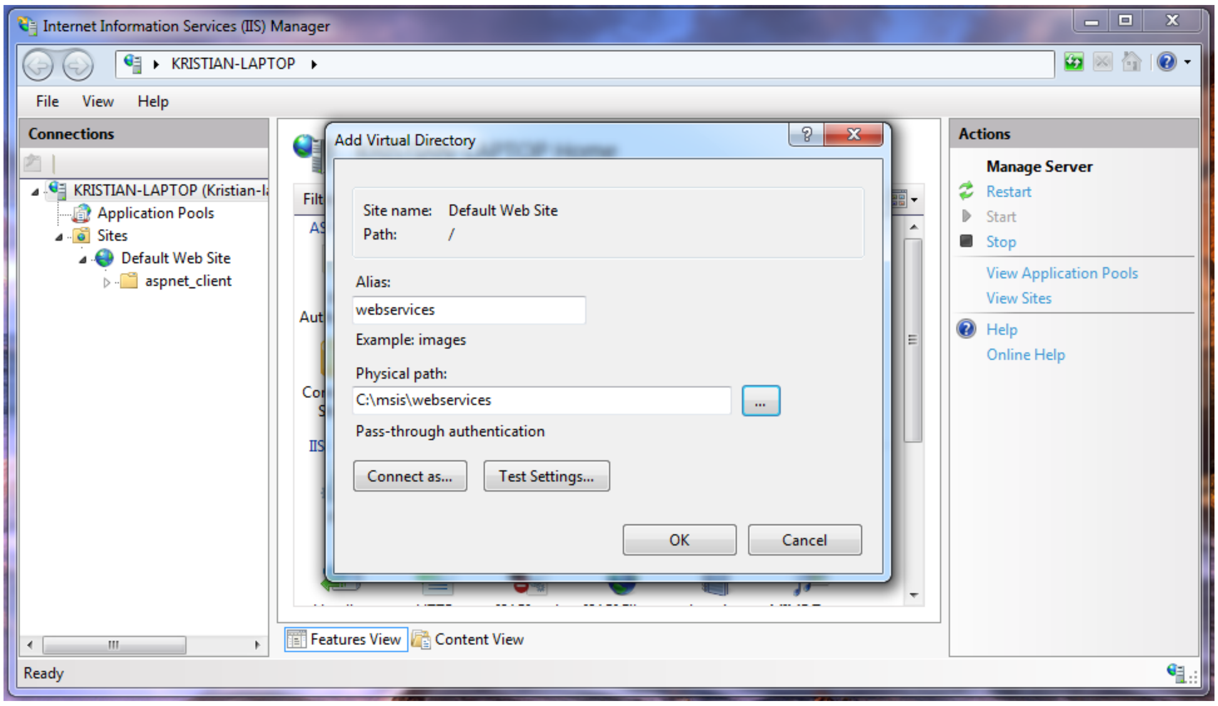

Site name: Default Web Site
Path: /
Alias: webservices
Example: images
Physical path: C:\msis\webservices
Pass-through authentication

Figure 7: Adding a Virtual Directory in IIS

When done, right click the newly created virtual directory and select ***Convert to Application***. Just press OK in the dialog that appears. Observe that the icon of your virtual directory changes from: into: .

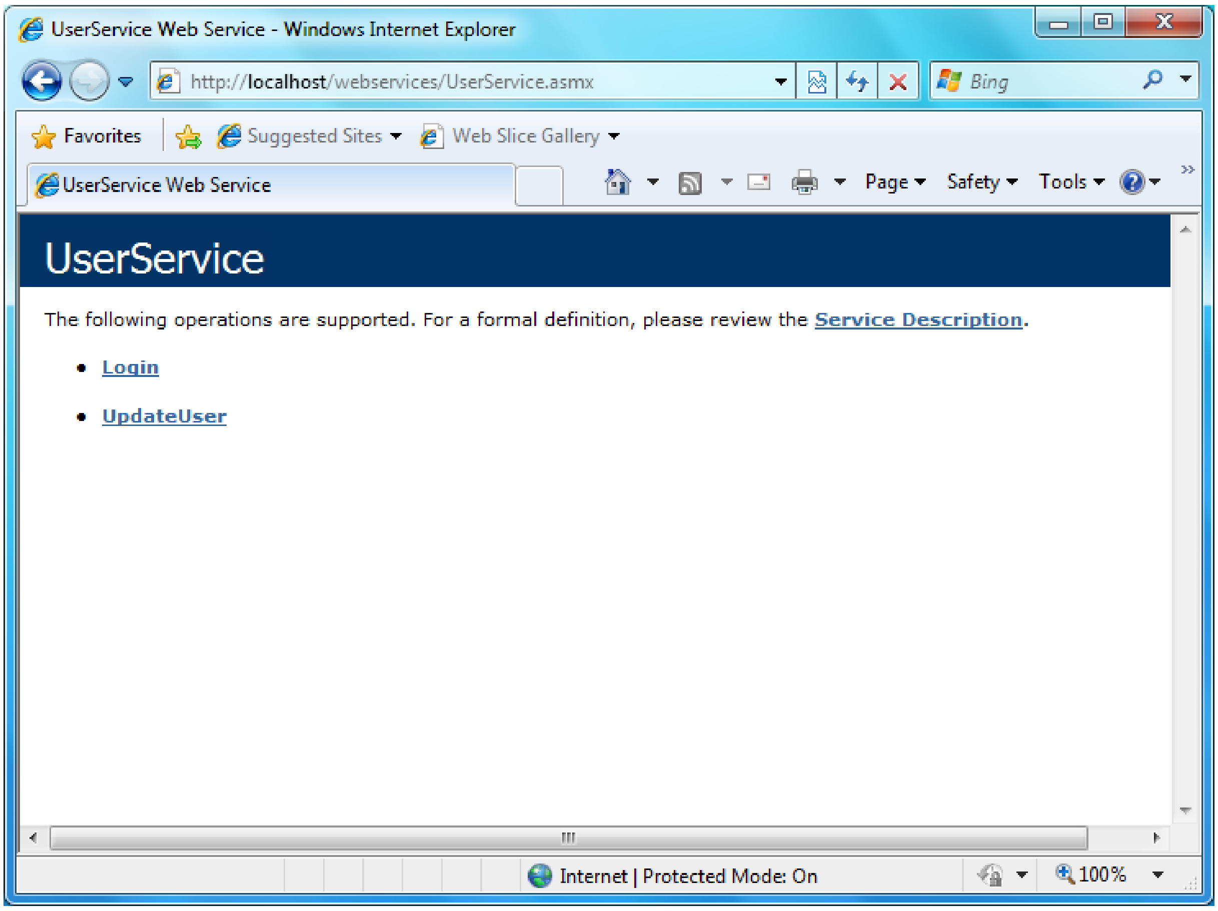
Now right click on the application and press on ***Explore***. Locate the file named ***SQLConn.cs*** in the folder named ***App_Code***, and open it in an editor (e.g. Visual Studio or simply Notepad). Locate the section in the file that looks like Figure 8. This string needs to be edited based on the settings from the previous section. Data source is the address for connecting to the server, written by default like ***COMPUTER-NAME\\sqlexpress***,***2301*** is the port number for connecting (Remember to allow TCP connections to this port if you haven't already in Firewall settings, e.g. under Exceptions in Windows Firewall). Initial Catalog is the database name, here defaulted to **msis**. User ID and Password are the username and password from the installation of SQL Server.

```
static string connectionString = "Data Source=Kristian-skole\\sqlexpress,2301;Initial Catalog=msis;"
                               + "User ID=sa;Password=msis;Connection Timeout=5";
```

Figure 8: Connection string for connecting code and database

After this has been done, the web services should be up and running. An easy way to test if it works is to open your browser of choice and navigate to ***http://localhost/the-name-you-selected-for-virtual-directory-in-fig-7/UserService.asmx***. If the result looks like Figure 9, then everything should be OK. In total there are 4 web services named:

- AnnouncementService.asmx
- LocationService.asmx
- ScheduleService.asmx
- UserService.asmx


UserService Web Service - Windows Internet Explorer
http://localhost/webservices/UserService.asmx
Bing
Favorites
Suggested Sites
Web Slice Gallery
UserService Web Service
Page
Safety
Tools
Internet | Protected Mode: On
100%

UserService

The following operations are supported. For a formal definition, please review the Service Description.

- Login
- UpdateUser

Figure 9: Example of running web service

You should test to see that they all work. The next section handles the schedule service, and how the address for this service can be changed.

Retrieving data from the schedule system

The web service uses an external source for retrieving schedule data and user course selection. At the time of writing (august 2009), this source was located at: http://ntnu.1024.no/. If or when this address is changed, one line of code also needs to be updated in the web services. The files that need to be updated are:

- ScheduleService.cs in App_Code folder
- AnnouncementService.cs in App_Code folder

Look for the string seen in Figure 10near the top of the files, and change the address if needed.

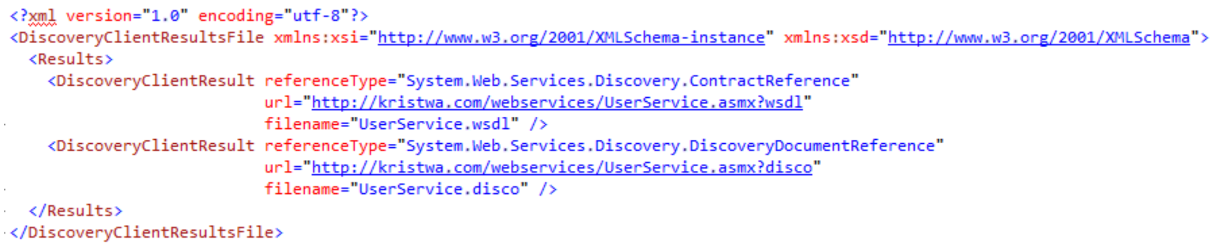
```
// ADRESS TO SCHEDULE SERVICE
string icalUrl = "http://ntnu.1024.no/";
```

Figure 10: Modifying the address for the scheduling service

User Administration Website

This is the component that provides a website allowing users to set their course attendance etc. The site is located in the same directory where the webservices folder is (see the previous section), and is named ***msis***. Open up ***Internet Information Services (IIS) Manager***and navigate to the Default Web Site in the left pane. Right click on Default Web Site and select ***Add Virtual Directory***. Give the virtual directory a name and set the physical path to the msis folder (e.g. C:\msis\msis). In the same manner as the previous section, convert this virtual directory to an application.

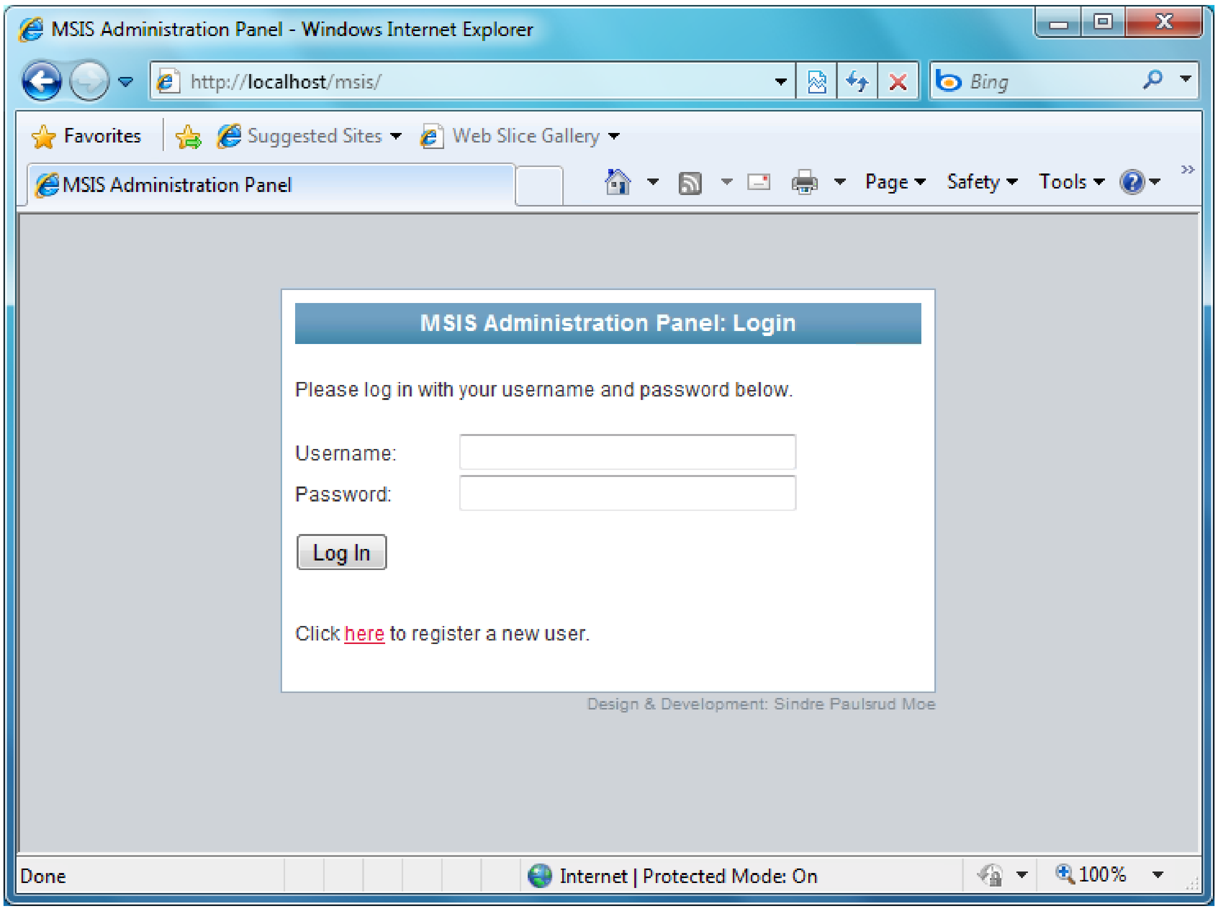
Open the project in Visual Studio and navigate in the ***msis*** part of the project to the folder named ***App_WebReferences***. The Web References needs to be updated. Specifically the url needs to be updated as seen in Figure 11.



```
<?xml version="1.0" encoding="utf-8"?>
<DiscoveryClientResultsFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Results>
    <DiscoveryClientResult referenceType="System.Web.Services.Discovery.ContractReference"
                           url="http://kristwa.com/webservices/UserService.asmx?wsdl"
                           filename="UserService.wsdl" />
    <DiscoveryClientResult referenceType="System.Web.Services.Discovery.DiscoveryDocumentReference"
                           url="http://kristwa.com/webservices/UserService.asmx?disco"
                           filename="UserService.disco" />
  </Results>
</DiscoveryClientResultsFile>
```

Figure 11: Web service XML connection string

Afterwards test in your browser that http://localhost/msis gives a site like in Figure 12.


MSIS Administration Panel - Windows Internet Explorer

http://localhost/msis/

MSIS Administration Panel: Login

Please log in with your username and password below.

Username:

Password:

Log In

Click here to register a new user.

Design & Development: Sindre Paulsrud Moe

Done | Internet | Protected Mode: On | 100%

Figure 12: MSIS Administration Panel

This concludes the server side configuration. The next chapter handles the client configuration.

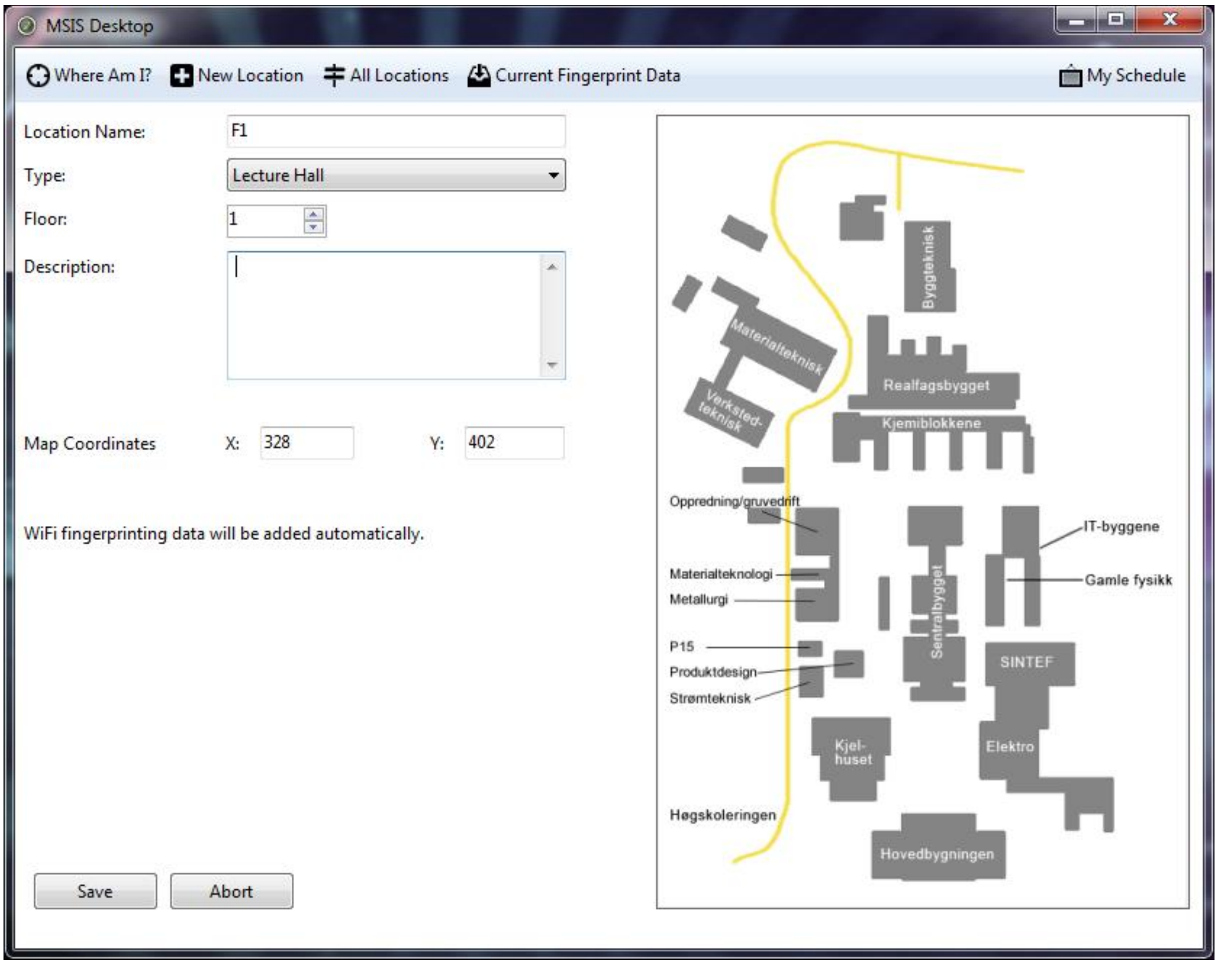
Client Configuration

This chapter contains information on configuring the client applications.

Desktop client

The desktop client is currently made to allow testing functionality that the web services exposes easier. A sample screenshot of the desktop client can be seen in Figure 13.

The client requires .NET Framework 3.5 to operate. Also, some features in the program will be disabled on machines with no WiFi adapters.



Location Name: F1

Type: Lecture Hall

Floor: 1

Description:

Map Coordinates X: 328 Y: 402

WiFi fingerprinting data will be added automatically.

Figure 13: MSIS Desktop client

Mobile client

The mobile client is currently designed to work exclusively on mobile phones running the Windows Mobile operating system. It has been tested primarily on devices running Windows Mobile 6.1, however briefly also on devices dating back to Windows Mobile 5.0.

Furthermore it requires the Microsoft .NET 3.5 Compact Edition framework. This can be installed either by:

- Connecting the device to a computer which has ActiveSync or Windows Mobile Device Center installed, and then running the .NET Compact Framework Installation Wizard (http://www.microsoft.com/downloads/details.aspx?FamilyID=e3821449-3c6b-42f1-9fd9-0041345b3385&displaylang=en).

OR

- Running the CAB installation directly on the mobile device (unpack the downloaded installation file from above).

Getting MSIS to run on the Mobile Phone

When the .NET Compact Framework has been installed on the device, MSIS is ready to be deployed to the device. Initially one could copy the CAB file from the project archive over to the phone and install it, but during further development it would be more beneficial to deploy to the device directly from Visual Studio.

In Visual Studio select Windows Mobile 6 Professional Device as target platform (See Figure 14). One could also download emulators for Windows Mobile 6.1 or Windows Mobile 6.5 if wanting to develop for even newer platforms (although 6.0 applications work fine on newer versions, but not necessarily the other way). Afterwards it should be just to build, deploy and run the application.

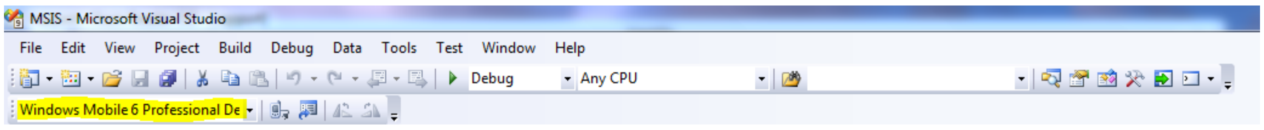

MSIS - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Test Window Help
Debug
Any CPU
Windows Mobile 6 Professional De

Figure 14: Selecting the proper target device

Important note: When first running the application, it's recommended to tap ***Settings*** and revise the URL addresses for the Web Services and Map Data. If you want to address these URL pointers in a more permanent matter, check ***AppSettings.cs*** and revise the values stored in there.

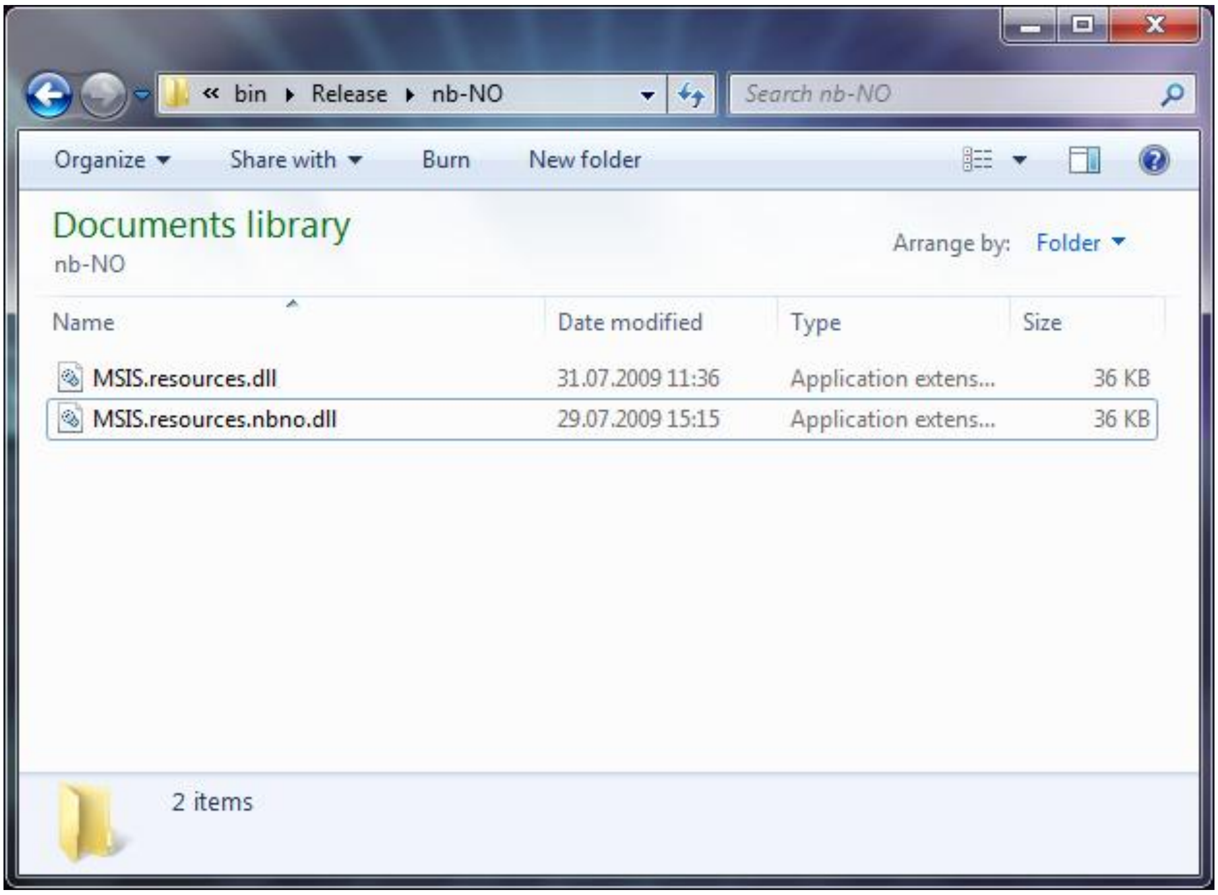
Creating Windows Mobile Installers

It proved to be quite a challenge to create a functioning installer for Windows Mobile. There is a bug in the utility that Visual Studio uses to create installers that doesn't allow any filenames to be the same in an installation package (ref: https://connect.microsoft.com/VisualStudio/feedback/ViewFeedback.aspx?FeedbackID=117453). The localization output from the client requires the filename to be MSIS.resources.dll. Therefore a workaround is needed.

The first step is to compile the client as usual. Then open windows explorer and navigate to the location of the compilation output. There should be two folders there, named ***en*** and ***nb-NO***. In these folders there are one file named MSIS.resources.dll. Rename or make a copy of this file in both folders to append the language name so that they become:

- MSIS.resources.en.dll
- MSIS.resources.nbno.dll

The application will look for these file names when starting, so it's important that these names are given. An example can be seen in Figure 15 below.



Name	Date modified	Type	Size
MSIS.resources.dll	31.07.2009 11:36	Application extens...	36 KB
MSIS.resources.nbno.dll	29.07.2009 15:15	Application extens...	36 KB

Figure 15: Renaming .dll files to achieve unique file names

Afterwards, open the installation project in Visual Studio, named ***CABProject***. Open the file system view and notice that there are two folders named ***en*** and ***nb-NO*** here also. The files in these folders need to be updated if the localization files have been updated. Simply remove the existing files there, right click, select Add -> File and navigate to the renamed .DLL files in the file browser that appears. Afterwards the installation project is ready to be compiled.

Note: This section only needs to be redone when the localization files have been edited.

Development environment

The MSIS client application and the web services have been developed in Microsoft Visual Studio2008. (This can at NTNU be provided from MSDNAA http://msdn60.e-academy.com/elms/Storefront/Home.aspx?campus=NO_700027. If you need a user for this, contact guru-tjenesten,http://guru.idi.ntnu.no) using the latest version of the .NET Framework (3.5). The client application is based on aWindows Mobile 6 Professional Smart Device project, while the web services are ASP.NET Web Service applications.In order to build the projects, the Microsoft .NET Framework 3.5 must first be installed.

The mobile clientapplication contains references to two third-party libraries; the OpenNETCF.AppSettings andOpenNETCF.Net libraries. The desktop client uses the Managed Wifi API. The web service code uses the DDay.iCal class library. All these external libraries are included in the mentioned projects that they are used in. For more information on these libraries, see the next section.

External libraries

The OpenNETCF Smart Device Framework (SDF) (http://www.opennetcf.com/) is a library of classes and application programming interfaces (APIs) which facilitates interaction with low-level functions part of the Windows Mobile operating system. Classes provided by the OpenNETCF.Net package are utilized in our implementation of the positioning module. The classes provide methods for network layer access and manipulation of the wireless communication facilities of the device. The MSIS client application makes use of the OpenNETCF SDF Community Edition, available for free and without registration on the website. The license allows for redistribution in both commercial and non-commercial projects.

DDay.iCal is an iCalendar class library written in C# and based on the RFC 2445 standard (http://www.ietf.org/rfc/rfc2445.txt). It parses files in the iCalendar format and provides an object-oriented interface to iCalendar components: Event, Todo, TimeZone, Journal, FreeBusy, and Alarm.DDay.iCal is available from the internet here: http://www.ddaysoftware.com/Pages/Projects/DDay.iCal/. It's licensed under the BSD license.

Managed Wifi API is a .NET class library allowing you to control Wifi (802.11) network adapters installed in your Windows machine programmatically. The library uses the Native Wifi API, which has been available since Windows Vista and Windows XP SP2 (limited support, requires hotfix available in KB 918997: http://support.microsoft.com/kb/918997). For more information on the Managed Wifi API check http://managedwifi.codeplex.com/. It's licensed under GNU Library General Public License (LGPL).

Project archive contents

Below is a table giving a small description of the contents of the project archive delivered with this document.

Folder	Subfolder	Description
Application_Server	maps	Map components (for webserver)
	msis	User administration website
	webservices	Web service components
CAB_Installer		Mobile client installer
Database		The MSIS database
Installation_Guide		This installation guide
MSIS_Desktop	bin	Compiled version of the desktop client
	src	Source code for desktop client
MSIS_Mobile	Client_Application	Client source code