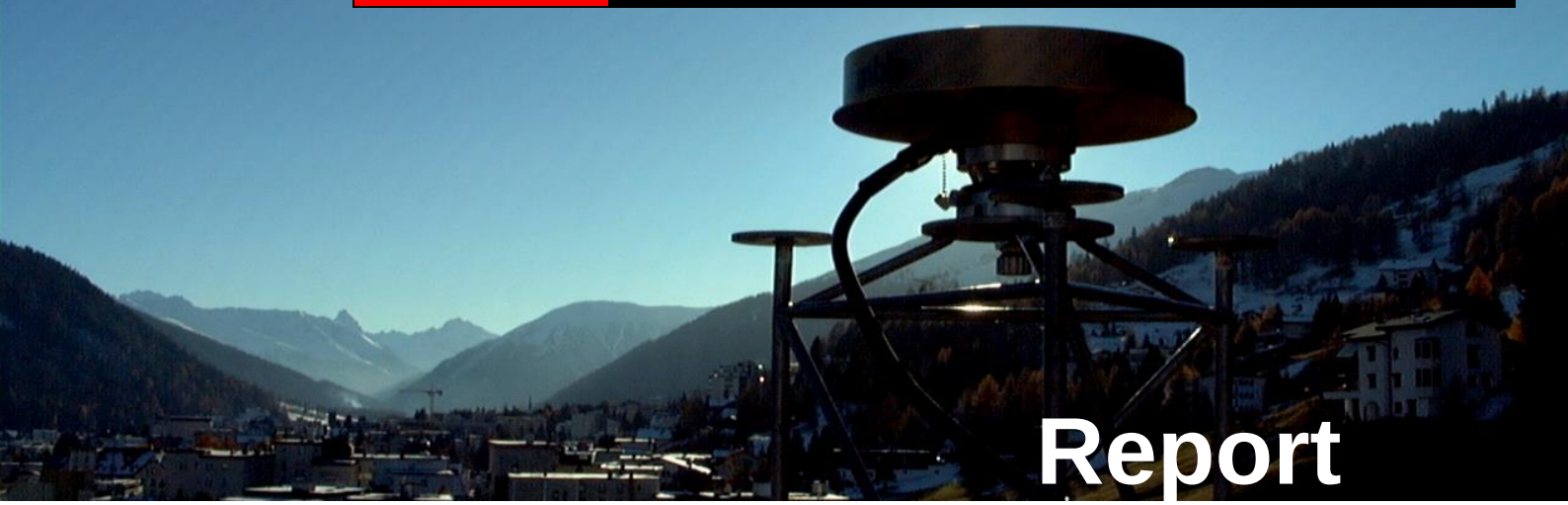

Report

REFRAME library

09-07 A

April 2016 Jérôme Ray Developer's manual

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Bundesamt für Landestopografie swisstopo
Office fédéral de topographie swisstopo
Ufficio federale di topografia swisstopo
Uffizi federal da topografia swisstopo

www.swisstopo.ch

SQS-Zertifikat ISO 9001:2000

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Federal Department of Defence,
Civil Protection and Sport

armasuisse

Federal Office of Topography swisstopo

REFRAME library

09-07 A

Jérôme Ray

Developer's manual

April 2016

Table of contents

Bundesamt für Landestopografie
Office fédéral de topographie
Ufficio federale di topografia
Uffizi federal da topografia
Federal Office of Topography

Redaktion: A. Wiget
Seftigenstrasse 264
CH-3084 Wabern
Tel: +41 31 963 24 69
Fax: +41 31 963 24 59
E-mail: adrian.wiget@swisstopo.ch

---

swisstopo Report ist die Nachfolgeserie der Reihe "Technische Berichte" in welcher über die wichtigen Arbeiten aus den Bereichen von swisstopo berichtet wird.
swisstopo Report est la suite de la série "Rapports techniques", qui décrivent les projets et activités les plus importants de swisstopo.
swisstopo Report is the follow-up of the series "Technical Report" in which important projects and activities of swisstopo are described.

1 Introduction

REFRAME is a planimetry and/or altimetry transformation software module[1] for technical surveying or cadastral surveying applications, with the highest precision requirements.

This product supports the following Swiss reference frames/systems:

- Plane coordinates LV03 (CH1903)
- Plane coordinates LV95 (CH1903+)
- Global geographic or geocentric coordinates CHTRS95/ETRS89/WGS84
- National levelling network LN02 (levelled heights)
- National height network LHN95 (orthometric heights, CHGeo2004)
- Ellipsoidal heights (Bessel or GRS80)

REFRAME can compute transformations in interactive mode (manual input of coordinates) or file mode (supports several formats like ESRI Shape, Interlis, DXF, swisstopo LTOP, Topobase, Excel CSV, text) and can also be used as batch tool in command line.

Since version 1.5 (January 2009), this software can also transform the metadata of raster files (World files, GeoTIFF or ECW).

Several calculation services (interactive, file transformation, HTTP REST) have been launched since 2007, too.

For more information about REFRAME desktop software or web services, please refer to our website:

→ *http://www.swisstopo.ch/geosoftware*

And for technical aspects about Swiss reference frames and the new frame LV95:

→ *http://www.swisstopo.ch/lv95*

The REFRAME library allows software developers to implement REFRAME computing in their own applications with .NET, COM or LIB components (.NET or C++ applications, Silverlight Web applications, VBA macros, further DLL or plug-ins, etc.) or a JAR archive (Java applications, Servlets, etc.).

A set of classes is also available for Java desktop and online applications.

For technical reasons, the computations are split in two functions/modules in this library:

- **REFRAME** (method “ComputeReframe”) for transformations between Swiss reference frames (planimetry and altimetry)
- **GPSREF** (method “ComputeGpsref”) for transformations between Swiss and global (European/World) coordinates (and reference ellipsoid change)

[1] The full product name is “REFRAME for GeoSuite”, please refer to http://www.swisstopo.ch/geosuite for more information

2 Technical aspects

The REFRAME library can compute planimetric and altimetric reference frame changes. All the transformations are not linear and are based on binary datasets organized in grids (altimetry) or triangular network (planimetry).

The following datasets are embedded:

- **CHENyx06 (FINELTRA)**

  The transformation from LV03 to LV95 coordinates and reverse is based on the FINELTRA algorithm[2] ("affine transformation with finite elements"). It is based on a triangular network transformation dataset named CHENyx06. Current version: 03.2007.

- **HTRANS**

  The transformation from LN02 (levelled) to LHN95 (orthometric) heights is based on the HTRANS transformation dataset. This transformation is based on 3 regular grids (with a resolution of 1km x 1km) containing data depending on the influence of the gravimetric field, the incidence of kinematics (alpine rising over 100 years) and distortions of LN02 (quality of the precise levelling 1864-91). Current version: 2005.

- **CHGeo2004**

  The transformation from LHN95 (orthometric) to Bessel (ellipsoidal) heights is based on the Swiss geoid model. This model is defined as a regular grid (with a resolution of 1km x 1km). Current version: 2004.

For the transformation from Swiss LV95 plane coordinates to global geographic or geocentric coordinates (CHTRS95/ETRS89/WGS84) or reverse, the rigorous projection formulas are used (**Oblique Mercator and "LV95" parameters**[3]). If you want to transform LV03 coordinates to global coordinates, you'll have to transform them to LV95 first, and then to ETRS89. The same intermediate step applies for the reverse computation: ETRS89 to LV95 and then to LV03.

3 Minimal system requirements

3.1 C#, Silverlight or C++/CLI DLL

- Operating system:
  - Microsoft Windows (at least XP SP3), Mac OS X or following Linux distribution[4]: Debian, Ubuntu, Fedora, CentOS, openSUSE, SLES or Gentoo
- Framework/runtime:
  - **Microsoft .NET Framework**[5] 3.5 and/or further: sometimes both 3.5 and 4.0 (or newer) are needed, at least when developing in C++ with Visual Studio 2010 (which is automatically targeting .NET 4.0)

    or **MONO 2.10** or further (3.8 recommended)
  - **Microsoft Visual C++ 2010 Redistributable Package** for use of the C++/CLI DLL

[2] More information about FINELTRA and the LV03-LV95 transformation : *http://www.swisstopo.ch/lv95*

[3] More information about the Swiss projection and "LV95" parameters : *http://www.swisstopo.admin.ch/internet/swisstopo/en/home/topics/survey/sys/refsys.html*

[4] Only for C# DLL, Mono platform is required (*http://www.mono-project.com/*)

[5] Microsoft .NET Framework is not included in the provided setup to limit the file size, but it can be downloaded for free on the Microsoft website (*http://www.microsoft.com/downloads/*)

3.2 Java archive (JAR)

- Operating system:
  - Microsoft Windows, Mac OS X, Linux, Solaris, refer to Java system requirements[6]
- Framework/runtime:
  - **Java 7.0 or 8.0**

4 Package contents

The REFRAME library package contains this manual, binaries and code samples.

Description of the archive directories:

- **Binaries**
  - **AnyCPU** Binaries (.NET) for both x86 and x64 architecture ("AnyCPU"). This version of the DLL is not optimized for a specific CPU type and will run on both platforms. Note that some third-party components could not be available for x64 and require a specific x86 compilation (e.g. some database engines).
  - **COM_version**
    - **Setup** Windows Installer (.MSI) files for x86 and x64 versions of the COM DLL. Both needed files (binaries) are copied to *%ProgramFiles%\swisstopo\ReframeDLL*, the type library file (.tlb) is generated and the required entries are added to the Windows registry.
    - **x64** Binaries (COM DLL) for x64 architecture. It doesn't depend on the Windows version but on the target platform of your development. Use this version if you are compiling a x64 application.
    - **x86** Binaries (COM DLL) for x86 architecture. It doesn't depend on the Windows version but on the target platform of your development. Use this version if you are compiling a x86 application. It will also work on x64 platforms.
  - **Java** Classes package (JAR) compiled for Java 1.6, but compatible with further versions.
  - **Silverlight** Binary (.NET) for Silverlight applications (client plug-in required).
  - **x64** Binaries (.NET and C++/CLI wrapper) for x64 architecture. It doesn't depend on the Windows version but on the target platform of your development. Use this version if you are compiling a x64 application.
  - **x86** Binaries (.NET and C++/CLI wrapper) for x86 architecture. It doesn't depend on the Windows version but on the target platform of your development. Use this version if you are compiling a x86 application. It will also work on x64 platforms.

- **Code_samples**
  - **Config_for_specific_path** Examples of ".exe.config" files which allow to move "swisstopo.reframelib.dll" and "swisstopo.data.dll" files to another location than the application (exe) directory (for .NET/C++ applications). Refer to paragraph 5.1 for more information about this.
  - **References** Required REFRAME binaries referenced in the various sample projects.

[6] Minimal system requirements for Java are available on the Java website: *http://java.com/sysreq*

- **ReframeSampleCpp** Native C++ example that uses the C++/CLI wrapper. No registry entry needed.
- **ReframeSampleCppCom** Native C++ example that uses the COM DLL. REFRAME DLL must have been previously registered in the system (Windows registry).
- **ReframeSampleCS** C# example (fully .NET application using C# REFRAME binaries)
- **ReframeSampleJava** Java example.
- **ReframeSampleSilverlight** Silverlight Web app example.
- **ReframeSampleVBA** VBA for Microsoft Excel example using the COM DLL. REFRAME DLL must have been previously registered in the system (Windows registry).

- **Documentation** This developer manual (published as swisstopo Report) in PDF format

5 Installation and deployment

5.1 Use of the managed DLL (for .NET applications)

You don't need any installation, except the copy of the two necessary DLL files somewhere on your computer or in your project directory (recommended):

- **swisstopo.reframelib.dll** (native C# functions)
- **swisstopo.data.dll** (transformation datasets)

When you compile your application, both files will be automatically copied to the output directory. If you do not want to have these both files in the application directory (where the main exe is), you can create a text file with the same name as the exe and with the ".config" extension, for example "myReframeApp.exe.config" if your application is named "myReframeApp.exe".

It's content must look like:

```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
        <probing privatePath="plugins\reframe"/></assemblyBinding>
   </runtime>
</configuration>
```

The attribute "privatePath" must contain the **relative path** to the both DLL files. These files must be "under" (in a subdirectory of) the main exe path.

If you want to choose a **fully independent directory**, you can use the following directive:

```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity name="swisstopo.reframelib"
culture="neutral" publicKeyToken="9ffb35f31e07cf2f"/>
            <codebase version="3.2.0.0"
href="FILE://C://MyPath//swisstopo.reframelib.dll"/>
         </dependentAssembly>
         <dependentAssembly>
            <assemblyIdentity name="swisstopo.data"  culture="neutral"
publicKeyToken="ea90158b1bff6de6"/>
            <codebase version="1.1.0.0"
href="FILE://C://MyPath//swisstopo.data.dll"/>
         </dependentAssembly>
```

```
      </assemblyBinding>
   </runtime>
</configuration>
```

Beware that the **full absolute path has to be specified here in both "href" attributes**. When you deploy your application, you will have to build a script or setup that writes these values correctly, relative to the target computer.

5.2 Use of the managed DLL (for ASP.NET web sites)

You don't need any installation, except the copy of the two necessary DLL files somewhere on the server or in your project directory (recommended):

- **swisstopo.reframelib.dll** (native C# functions)
- **swisstopo.data.dll** (transformation datasets)

Beware that when "shadow copying" technique is enabled, the assemblies are copied form the application directory into a temporary folder (cache). That means that all the DLL files will not be in the same folder any more at runtime, and a "FileNotFoundException" will probably occur because the both files above cannot be found any more.

The easiest way to avoid that is to disable shadow copying. This can be done in the "Web.config" file:

```
<configuration>
   <system.web>
      <hostingEnvironment shadowCopyBinAssemblies="false" />
   </system.web>
</configuration>
```

If you want to choose a **fully independent directory** or if you cannot or don't want to disable the shadow copying, you can use the following directive:

```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <dependentAssembly>
            <assemblyIdentity name="swisstopo.reframelib"
culture="neutral" publicKeyToken="9ffb35f31e07cf2f"/>
            <codebase version="3.2.0.0"
href="http://www.mywebsite.com/myprojectfolder/swisstopo.reframelib.dll"/>
         </dependentAssembly>
         <dependentAssembly>
            <assemblyIdentity name="swisstopo.data"  culture="neutral"
publicKeyToken="ea90158b1bff6de6"/>
            <codebase version="1.1.0.0" href="
http://www.mywebsite.com/myprojectfolder/swisstopo.data.dll"/>
         </dependentAssembly>
      </assemblyBinding>
   </runtime>
</configuration>
```

5.3 Use of the C++/CLI DLL (for C++ applications)

You don't need any installation, except a copy of the four necessary DLL files somewhere on your computer or in your build directory (recommended):

- **swisstopo.reframelib.dll** (native C# functions)
- **swisstopo.data.dll** (transformation datasets)
- **ReframeLibWrapper.dll** (C++/CLI wrapper functions)
- **ReframeLibWrapper.lib** (Linker input / import library)

In order to compile your application, "ReframeWrapper.lib" must be accessible by the linker (refer to chapter 7 for details and example) and the three DLL files must be found in the application directory

("debug" directory after compilation). You unfortunately do have no choice for the "ReframeLibWrapper.dll", unless you rewrite it yourself and integrate it directly in your application (which has to be C++/CLI with Common Language RunTime Support). Both "swisstopo.reframelib.dll" and "swisstopo.data.dll" can be placed in another directory: you have to create a text file with the same name as your exe and with the extension ".config", for example "myReframeApp.exe.config" if your application is named "myReframeApp.exe".

It's content must look like explained in preceding paragraph 5.1.

5.4 Use of the COM DLL (for all applications)

swisstopo delivers a Windows Installer setup which installs all required files and automatically registers the DLL in the system. You just have to execute it and follow the onscreen instructions.

To optimize file size and download time, we didn't integrate neither the Microsoft Windows Installer engine nor the Microsoft Windows .NET Framework. If they aren't already installed on your computer, they will be automatically downloaded[7] and installed.

If you want to create your own distribution or install the DLL on another computer without our setup, or if you are experiencing problems with our setup (e.g. Reframe object cannot be instantiated), you'll have to do as follows:

1. Copy the program library file into the chosen installation directory:
   - **swisstopoReframeLib.dll**
2. Register the .NET assembly in the Windows registry by running the following command:

   regasm swisstopoReframeLib.dll /codebase /tlb

   *Warning: on systems where Microsoft .NET Framework SDK (or Visual Studio) isn't installed (but only the .NET Framework redistributable), you'll have to type the full "regasm" path, e.g. "C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\regasm.exe …". You will need to specify explicitly "…\Framework**64**\..." for the 64-bit version.*

   Warning: COM components are identified by unique GUIDs. A component (GUID) can only be registered once on a target system. This means that parallel registrations are not possible. If you install the REFRAME DLL twice in two different directories, the latest installed one will remain in the Windows registry. It will work, but only one DLL will be used for both applications.

   A problem can be encountered if a relative path is specified on the registry (what should never be done) or if the latest version is uninstalled and the registry entries are not updated. This will never happen if you always use the official DLL setup delivered.

   Note: this registration can be done automatically with most of the setup packaging software, as they support natively the "COM Interop" registration.
3. Copy the datasets resource file in the same directory as "swisstopoReframeLib.dll":
   - **swisstopo.data.dll**

5.5 Use of the Silverlight DLL (for Web applications)

You don't need any installation, except the copy of the necessary DLL file in your project directory:

- **swisstopo.reframelibSl.dll** (native C# functions and transformation datasets)

When you compile your application, this file will be automatically copied to the output directory.

On the client computer (final user), the free Silverlight **plug-in**, only available for some Web browsers, will be required.

[7] They also can be obtained through the Windows Update service or downloaded manually : http://www.microsoft.com/downloads

5.6 Use of the Java classes (for Java desktop or Web applications)

You don't need any installation, except an import of the provided JAR file in your project:

- **reframeLib.jar** (native Java functions and transformation datasets)

This Java archive contains all required classes and also the transformation datasets (binary files that are embedded in "swisstopo.data.dll" for .NET version).

For a further deployment of our application, you will have to distribute this JAR file with your application and add it to the **classpath**.

6 Using the managed DLL in .NET applications (e.g. C#)

6.1 Main class

The REFRAME DLL main class is named "**Reframe**" and is defined in the namespace "**swisstopo.geodesy.reframe**".

The only one constructor takes no argument:

```
public Reframe()
```

6.2 Distinct types (enum)

- **PlanimetricFrame** (for use with "ComputeReframe" function):
  - **LV03_Military** = 0
    Swiss plane (military) coordinates LV03 (CH1903), EPSG code 21781
  - **LV95** = 1
    Swiss plane coordinates LV95 (CH1903+), EPSG code 2056
  - **LV03_Civil** = 2
    Swiss civil coordinates LV03C-G (CH1903), EPSG code 21782
- **AltimetricFrame** (for use with "ComputeReframe" function):
  - **LN02 = 0**
    National levelling network LN02 (levelled heights)
  - **LHN95 = 1**
    National height network LHN95 (orthometric heights on geoid "CHGeo2004")
  - **Ellipsoid = 2**
    Ellipsoidal heights on Bessel 1841 (CH1903/CH1903+) or GRS80 (ETRS89)
  - **CHGeo98 = 3**
    Preliminary orthometric heights (on geoid "CHGeo98")
- **ProjectionChange** (for use with "ComputeGpsref" function):
  - **ETRF93GeocentricToLV95** = 0
    Transformation of geocentric coordinates ETRF93 (ETRS89) [m] into Swiss plane coordinates LV95 (CH1903+) [m]
  - **ETRF93GeographicToLV95** = 1
    Transformation of geographic coordinates ETRF93 (ETRS89) [°] into Swiss plane coordinates LV95 (CH1903+) [m]
  - **LV95ToETRF93Geocentric** = 2
    Transformation of Swiss plane coordinates LV95 (CH1903+) [m] into geocentric coordinates ETRF93 (ETRS89) [m]
  - **LV95ToETRF93Geographic** = 3
    Transformation of Swiss plane coordinates LV95 (CH1903+) [m] into geographic coordinates ETRF93 (ETRS89) [m]

6.3 Methods and returns

- **ComputeGpsref**:
  Transformation of Swiss plane coordinates LV95 and ellipsoidal heights on Bessel to global geographic or geocentric CHTRS95/ETRS89/WGS84 coordinates and ellipsoidal heights on GRS80, or reverse.
- **ComputeReframe**:
  Transformation of Swiss plane coordinates LV03 to LV95 or reverse, and transformations between Swiss levelled heights LN02, orthometric heights LHN95 and ellipsoidal heights on Bessel.

ComputeGpsref

This method takes 4 arguments:

```
void ComputeGpsref(ref double east_x_lon, ref double north_y_lat,
                   ref double height_z,
                   ProjectionChange transformation);
```

1. **east_x_lon:**
   Swiss LV95 easting coordinate in meters [m], CHTRS95/ETRS89/WGS84 geocentric X coordinate in meters [m] or CHTRS95/ETRS89/WGS84 geographic longitude (λ) in decimal degrees [°]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
2. **north_y_lat:**
   Swiss LV95 northing coordinate in meters [m], CHTRS95/ETRS89/WGS84 geocentric Y coordinate in meters [m] or CHTRS95/ETRS89/WGS84 geographic latitude (φ) in decimal degrees [°]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
3. **height_z:**
   Ellipsoidal Height in meters [m], on Bessel 1841 if input values are Swiss plane coordinates in LV95, on GRS80 if input values are global coordinates in CHTRS95/ETRS89/WGS84. This argument is passed by reference, which means that the input value is replaced by the output height after the processing.
4. **transformation:**
   A constant (distinct type) that defines the transformation to compute (input and output reference frames). Refer to paragraph 6.2, under "**ProjectionChange**", for possible values.

This method does not return any value.

It something fails, an exception is thrown. For example:

- ArgumentOutOfRangeException: Input coordinates are not LV95 (transformation from LV95 to ETRF93)
- Exception: General unmanaged exception. It should never happen, but it is strongly recommended to catch it to avoid unexpected results, for example because of an incompatible system.

The exception message gives more information about the error.

ComputeReframe

This method takes 7 arguments:

```
bool ComputeReframe(ref double east, ref double north, ref double height,
                    PlanimetricFrame plaFrameIn, PlanimetricFrame plaFrameOut,
                    AltimetricFrame altFrameIn, AltimetricFrame altFrameOut)
```

1. **east:**
   Easting coordinate in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
2. **north:**
   Northing coordinate in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
3. **height:**
   Height in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output height after the processing.
4. **plaFrameIn:**
   The planimetric reference frame of your source data. Refer to paragraph 6.2, under "**PlanimetricFrame**", for possible values.
5. **plaFrameOut:**
   The planimetric reference frame of your destination data. Refer to paragraph 6.2, under "**PlanimetricFrame**", for possible values.
6. **altFrameIn:**
   The altimetric reference frame of your source data. Refer to paragraph 6.2, under "**AltimetricFrame**", for possible values.
7. **altFrameOut:**
   The altimetric reference frame of your destination data. Refer to paragraph 6.2, under "**AltimetricFrame**", for possible values.

This method returns a boolean value:

- **true**: Transformation successfully executed with the standard method (CHENyx06/FINELTRA) without any exception. Input point is in the official Swiss TLM[8] perimeter.
- **false**: Transformation performed with a constant shift of +/- 2'000'000.000 m east and +/- 1'000'000.000 m north, because the input coordinates are outside the official Swiss TLM perimeter.

It something fails, an exception is thrown. For example:

- ArgumentOutOfRangeException: Altimetry could not be transformed, because input coordinates are outside HTRANS or CHENyx06 official Swiss TLM perimeter.
- FileNotFoundException: Transformation dataset files resource not found. Check if the file "swisstopo.data.dll" is available.
- FileLoadException: Transformation dataset files resource could not be loaded. Check if the file "swisstopo.data.dll" is available and valid.
- Exception: General unmanaged exception. It should never happen, but it is strongly recommended to catch it to avoid unexpected results, for example because of an incompatible system.

The exception message gives more information about the error.

[8] Topographic Landscape Model. Extent: emin=460'000 nmin=42'000 emax=868'000 nmax=322'000

Note: for a planimetric transformation LV03↔LV95, if the input point is outside the CHENyx06 triangular network (= outside the official swiss TLM perimeter), no exception is thrown because the transformation is officially defined as a translation of +/- 2'000'000 m and +/- 1'000'000 m. **You can get this information with the return value however**.

6.4 Code sample in C# .NET

Before you can use the DLL, you have to add a reference to "**swisstopo.reframelib.dll**" in your .NET project. The namespace is "swisstopo.geodesy.reframe".

Example of C# code:

```
// Reference to the REFRAME DLL
using swisstopo.geodesy.reframe;
…

// Instantiate a Reframe object
Reframe reframeObj = new Reframe();

// Input coordinates: read in a file, got from a textbox,
// or obtained through another method or library...
double c1 = 601000.0; // East in LV03
double c2 = 197500.0; // North in LV03
double c3 =    555.0; // Usual height (LN02)

// Transform LV03 coordinates to LV95 and LN02 height to Bessel
try
{
    bool outsideChenyx06 =
          !reframeObj.ComputeReframe(ref c1, ref c2, ref c3,
                                     Reframe.PlanimetricFrame.LV03_Military,
                                     Reframe.PlanimetricFrame.LV95,
                                     Reframe.AltimetricFrame.LN02,
                                     Reframe.AltimetricFrame.Ellipsoid);

    // If outsideChenyx06==true a warning must perhaps be done!

    // c1, c2 and c3 now contain transformed coordinates...
    … // TODO
}
catch (ArgumentOutOfRangeException e)
{
    // Coordinates outside Swiss TLM perimeter:
    // => height transformation impossible
    … // TODO
}
catch (FileNotFoundException e)
{
    // swisstopo.data.dll not found in app/dll directory
    … // TODO
}
catch (FileLoadException e)
{
    // swisstopo.data.dll could not be read or is corrupted
    … // TODO
}
```

```
catch (Exception)
{
     // other exception
    … // TODO
}
```

After this code, c1 (east in LV95) = 2601000.030 m, c2 (north in LV95) = 1197500.037 m and c3 (ellipsoidal height on Bessel) = 554.335 m.

```
// Succession of the first part above

// Transform LV95 coordinates to ETRS89 longitude/latitude
// and ellipsoidal height on Bessel to GRS80
try
{
     reframeObj.ComputeGpsref(ref c1, ref c2, ref c3,
                  Reframe.ProjectionChange.LV95ToETRF93Geographic);

    // c1, c2 and c3 now contain transformed coordinates...
    … // TODO
}
catch (Exception)
{
    // exception
}
```

After this code, c1 (longitude) = 7.451764 °, c2 (latitude) = 46.928595 ° and c3 (ellipsoidal height on GRS80) = 604.004 m.

7 Using the C++/CLI DLL in .C++ applications

7.1 Main class

The REFRAME main class is the same in C++ as in .NET: a C++/CLI wrapper has been written to allow access from a native C++ application.

Refer to chapter 0 above for more information about the .NET library.

7.2 Wrapper functions and import library

The C# DLL can be accessed from C++ through a C++/CLI[9] wrapper. To be able to use this component in a C++ application, you have to link the file “ReframeLibWrapper.lib”. Details and examples can be found below (paragraph 7.6).

The wrapper class is named “ReframeWrapper”.

7.3 Native header

To be able to access REFRAME types and methods, you need a C++ header which links the C++/CLI component (see above) and contains distinct types (constants) and class and methods declarations.

An example file is delivered with the REFRAME DLL package an is also printed below (paragraph 7.6).

[9] “Common Language Infrastructure” is a language specification intended to supersede Managed extensions for C++, which allowed C++ code to be targeted by the Common Language Runtime (CLR). CLR is the virtual machine component of Microsoft’s .NET framework.

7.4 Distinct types (enum)

- **ReframeWrapper**
  - Planimetric frames for use with “ComputeReframe” function:
    - **LV03_Military** = 0
      Swiss plane (military) coordinates LV03 (CH1903), EPSG code 21781
    - **LV95** = 1
      Swiss plane coordinates LV95 (CH1903+), EPSG code 2056
    - **LV03_Civil** = 2
      Swiss civil coordinates LV03C-G (CH1903), EPSG code 21782
  - Altimetric frames for use with “ComputeReframe” function:
    - **LN02** = 0
      National levelling network LN02 (levelled heights)
    - **LHN95** = 1
      National height network LHN95 (orthometric heights on geoid “CHGeo2004”)
    - **Ellipsoid** = 2
      Ellipsoidal heights on Bessel 1841 (CH1903/CH1903+) or GRS80 (ETRS89)
    - **CHGeo98** = 3
      Preliminary orthometric heights (on geoid “CHGeo98”)
  - Local↔global transformations for use with “ComputeGpsref” function:
    - **ETRF93GeocentricToLV95** = 0
      Transformation of geocentric coordinates ETRF93 (ETRS89) [m] into Swiss plane coordinates LV95 (CH1903+) [m]
    - **ETRF93GeographicToLV95** = 1
      Transformation of geographic coordinates ETRF93 (ETRS89) [°] into Swiss plane coordinates LV95 (CH1903+) [m]
    - **LV95ToETRF93Geocentric** = 2
      Transformation of Swiss plane coordinates LV95 (CH1903+) [m] into geocentric coordinates ETRF93 (ETRS89) [m]
    - **LV95ToETRF93Geographic** = 3
      Transformation of Swiss plane coordinates LV95 (CH1903+) [m] into geographic coordinates ETRF93 (ETRS89) [m]

7.5 Methods and returns

- **ComputeGpsref**:
  Transformation of Swiss plane coordinates LV95 and ellipsoidal heights on Bessel to global geographic or geocentric CHTRS95/ETRS89/WGS84 coordinates and ellipsoidal heights on GRS80, or reverse.
- **ComputeReframe**:
  Transformation of Swiss plane coordinates LV03 to LV95 or reverse, and transformations between Swiss levelled heights LN02, orthometric heights LHN95 and ellipsoidal heights on Bessel.

ComputeGpsref

This method takes 4 arguments:

```
void ComputeGpsref(ref double east_x_lon, ref double north_y_lat,
                   ref double height_z,
                   ProjectionChange transformation);
```

1. **east_x_lon:**
   Swiss LV95 easting coordinate in meters [m], CHTRS95/ETRS89/WGS84 geocentric X coordinate in meters [m] or CHTRS95/ETRS89/WGS84 geographic longitude (λ) in decimal degrees [°]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
2. **north_y_lat:**
   Swiss LV95 northing coordinate in meters [m], CHTRS95/ETRS89/WGS84 geocentric Y coordinate in meters [m] or CHTRS95/ETRS89/WGS84 geographic latitude (φ) in decimal degrees [°]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
3. **height_z:**
   Ellipsoidal Height in meters [m], on Bessel 1841 if input values are Swiss plane coordinates in LV95, on GRS80 if input values are global coordinates in CHTRS95/ETRS89/WGS84. This argument is passed by reference, which means that the input value is replaced by the output height after the processing.
4. **transformation:**
   A constant (distinct type) that defines the transformation to compute (input and output reference frames). Refer to paragraph 6.2, under "**ProjectionChange**", for possible values.

This method does not return any value.

It something fails, a runtime error (std::runtime_error) which contains an error description is thrown.

It is strongly recommended to catch runtime errors to avoid unexpected results.

The runtime error message ("what()" method) gives more information about the error.

ComputeReframe

This method takes 7 arguments:

```
bool ComputeReframe(ref double east, ref double north, ref double height,
                    PlanimetricFrame plaFrameIn, PlanimetricFrame plaFrameOut,
                    AltimetricFrame altFrameIn, AltimetricFrame altFrameOut)
```

1. **east:**
   Easting coordinate in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
2. **north:**
   Northing coordinate in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
3. **height:**
   Height in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output height after the processing.
4. **plaFrameIn:**
   The planimetric reference frame of your source data. Refer to paragraph 6.2, under "**PlanimetricFrame**", for possible values.
5. **plaFrameOut:**
   The planimetric reference frame of your destination data. Refer to paragraph 6.2, under "**PlanimetricFrame**", for possible values.
6. **altFrameIn:**
   The altimetric reference frame of your source data. Refer to paragraph 6.2, under "**AltimetricFrame**", for possible values.

7. **altFrameOut:**
   The altimetric reference frame of your destination data. Refer to paragraph 6.2, under "**AltimetricFrame**", for possible values.

This method returns a boolean value:

- **true**: Transformation successfully executed with the standard method (CHENyx06/FINELTRA) without any exception. Input point is in the official swiss TLM perimeter.
- **false**: Transformation performed with a constant shift of +/- 2'000'000.000 m east and +/- 1'000'000.000 m north, because the input coordinates are outside the official swiss TLM perimeter.

It something fails, a runtime error (std::runtime_error) which contains an error description is thrown.

It is strongly recommended to catch runtime errors to avoid unexpected results.

The runtime error message ("what()" method) gives more information about the error.

Note: for a planimetric transformation LV03↔LV95, if the input point is outside the CHENyx06 triangular network (= outside the official swiss TLM perimeter), no runtime error is thrown because the transformation is officially defined as a translation of +/- 2'000'000 m and +/- 1'000'000 m. ***You can get this information with the return value however****.*

If you try to compute both planimetric and altimetric transformation, you will get a runtime error, because the height transformation is undefined for a point which is outside the transformation grids.

7.6 Code sample in native C++

Before you can use the DLL, you have to link the "**swisstopo.reframelib.dll**". First of all, "swisstopo.reframelib.lib" must be found in the debug output directory or its path must be given to the linker. In Microsoft Visual Studio, this can be done for example in the project properties, under "**Configuration Properties > Linker > General**". Add the path under "**Additional Library Directories**".

In a header file, you will then be able to link the REFRAME library with a pragma comment:

```
#pragma comment(lib, "ReframeLibWrapper.lib")
```

In the same header, you can either link the C+++ ReframeWrapper header which is delivered with the other binaries or recreate it by yourself.

Example of C++ header (e.g. "myHeader.h") with REFRAME references:

```
#include "ReframeWrapper.h"
#pragma comment(lib, "ReframeLibWrapper.lib")
```

Example of C++ header (e.g. "myHeader.h") with LIB link and DLL import in the same file:

```
#pragma comment(lib, "ReframeLibWrapper.lib")

class __declspec(dllimport) ReframeWrapper
{
```

```
public:
   // Enums / reference frames
   enum PlanimetricFrame
   {
         LV03_Military = 0,
         LV95 = 1,
         LV03_Civil = 2,
   };
   enum AltimetricFrame
   {
         LN02 = 0,
         LHN95 = 1,
         Ellipsoid = 2,
         CHGeo98 = 3
   };
   enum ProjectionChange
   {
         ETRF93GeocentricToLV95 = 0,
         ETRF93GeographicToLV95 = 1,
         LV95ToETRF93Geocentric = 2,
         LV95ToETRF93Geographic = 3
   };

   // Initialization
   ReframeWrapper();
   virtual ~ReframeWrapper();

   // Methods
   bool ComputeReframe(double &east, double &north,
         double &height, PlanimetricFrame plaFrameIn,
         PlanimetricFrame plaFrameOut, AltimetricFrame altFrameIn,
         AltimetricFrame altFrameOut);
   void ComputeGpsref(double &east_x_lon, double &north_y_lat,
         double &height_z, ProjectionChange transformation);

private:
   // Pointer to REFRAME library object
   void* m_pReframeClr;
};
```

Example of C++ source file:

```
// Header containing REFRAME link/interface
#include "myHeader.h[10]"

// Create a REFRAME object
ReframeWrapper reframeLibObj;

// Input coordinates: read in a file, got from a textbox,
// or obtained through another method or library...
double c1 = 601000.0; // East in LV03
double c2 = 197500.0; // North in LV03
double c3 =    555.0; // Usual height (LN02)
```

[10] The header file must be defined in the form of one of the both header samples above

```
// Transform LV03 coordinates to LV95 and LN02 height to Bessel
try
{
    bool outsideChenyx06 = !reframeLibObj.ComputeReframe(
              c1, c2, c3,
              ReframeWrapper::LV03_Military, ReframeWrapper::LV95,
              ReframeWrapper::LN02, ReframeWrapper::Ellipsoid);

    // c1, c2 and c3 now contain transformed coordinates...
    … // TODO
}
catch (std::runtime_error ex)
{
    // error: refer to ex.what() for more information
    … // TODO
}
catch (…)
{
    // unmanaged error, e.g. DLL cannot be loaded
    … // TODO
}
```

After this code, c1 (east in LV95) = 2601000.030 m, c2 (north in LV95) = 1197500.037 m and c3 (ellipsoidal height on Bessel) = 554.335 m.

```
// Succession of the first part above


// Transform LV95 coordinates to ETRS89 longitude/latitude
// and ellipsoidal height on Bessel to GRS80
try
{
    reframeLibObj.ComputeGpsref(c1, c2, c3,
              ReframeWrapper:: ETRF93GeographicToLV95);

    // c1, c2 and c3 now contain transformed coordinates...
    … // TODO
}
catch (std::runtime_error ex)
{
    // error: refer to ex.what() for more information
    … // TODO
}
```

After this code, c1 (longitude) = 7.451764 °, c2 (latitude) = 46.928595 ° and c3 (ellipsoidal height on GRS80) = 604.004 m.

8 Using the COM version

8.1 Introduction

Microsoft COM (Component Object Model) technology enables software components to communicate on Windows operating systems. COM objects can be created and used with a variety of programming languages. For example, the REFRAME COM DLL has been tested with the following programming languages:

- C#[11] (Microsoft Visual Studio.NET)
- C++ (Microsoft Visual Studio)
- Visual Basic for Applications (Microsoft Office)
- Pascal (Borland Delphi)
- Python (Python for .NET[12])

8.2 Warning

This version of the REFRAME DLL is pretty old and has initially been developed with minimal techniques and data types to ensure and maximal compatibility with applications, programming languages and architectures. The main advantage is that the library is easy to use, no special declaration or header are needed and there is no special prerequisite (except Microsoft .NET framework).
A big disadvantage is that distinct types/enumerations have not been used and the **function calls have to be done with codes (integer flags)**. It is the same for the function returns: **an integer code is given back to indicate if the processing was ok or not. No exception is throw**.

8.3 Main class

The REFRAME DLL main class is named "**Reframe**" and is accessible through an interface named "**IReframe**".

The only one constructor takes no argument:

```
public Reframe()
```

8.4 Methods and returns

The "reframeLib" class contains three public methods:

- **ComputeGpsref**:
  Transformation of Swiss plane coordinates LV95 and ellipsoidal heights on Bessel to global geographic or geocentric CHTRS95/ETRS89/WGS84 coordinates and ellipsoidal heights on GRS80, or reverse.
- **ComputeReframe**:
  Transformation of Swiss plane coordinates LV03 to LV95 or reverse, and transformations between Swiss levelled heights LN02, orthometric heights LHN95 and ellipsoidal heights on Bessel.
- **SetDatasetsDir**:
  This method has been deprecated and should not be used any more, except on very special configurations or for test purposes.

ComputeGpsref

This method needs 4 arguments:

```
int ComputeGpsref(ref double east_x_lon, ref double north_y_lat,
                  ref double height_z,
                  int flag);
```

[11] It is not really useful and recommended to work with COM in .NET, because the REFRAME DLL is a C# native library and can be accessed directly in .NET environment.

[12] Reframe class can also be accessed directly without COM, because the REFRAME DLL is a C# native library

1. **east_x_lon:**
   Swiss LV95 easting coordinate in meters [m], CHTRS95/ETRS89/WGS84 geocentric X coordinate in meters [m] or CHTRS95/ETRS89/WGS84 geographic longitude (λ) in decimal degrees [°]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
2. **north_y_lat:**
   Swiss LV95 northing coordinate in meters [m], CHTRS95/ETRS89/WGS84 geocentric Y coordinate in meters [m] or CHTRS95/ETRS89/WGS84 geographic latitude (φ) in decimal degrees [°]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
3. **height_z:**
   Ellipsoidal Height in meters [m], on Bessel 1841 if input values are Swiss plane coordinates in LV95, on GRS80 if input values are global coordinates in CHTRS95/ETRS89/WGS84. This argument is passed by reference, which means that the input value is replaced by the output height after the processing.
4. **flag:**
   A code that defines the transformation to compute (input and output reference frames). Available choices:
   - 0: Global geocentric coordinates (CHTRS95/ETRS89/WGS84) and height on GRS80 → Swiss plane coordinates LV95 (CH1903+) and height on Bessel 1841
   - 1: Global geographic coordinates (CHTRS95/ETRS89/WGS84) and height on GRS80 → Swiss plane coordinates LV95 (CH1903+) and height on Bessel 1841
   - 2: Swiss plane coordinates LV95 (CH1903+) and height on Bessel 1841 → Global geocentric coordinates (CHTRS95/ETRS89/WGS84) and height on GRS80
   - 3: Swiss plane coordinates LV95 (CH1903+) and height on Bessel 1841 → Global geographic coordinates (CHTRS95/ETRS89/WGS84) and height on GRS80

This method returns an integer value, a code which reports if the transformation was done successfully, or describes the error if not:

- 1: Computation successfully executed: the coordinates passed by reference have been updated with the new output values
- -1: Error: coordinates are outside the official Swiss TLM[13] perimeter (invalid input coordinates)
- -2: Error: unsupported value for “flag” argument (only “0”, “1”, “2” or “3” are allowed)

It is strongly recommended to check this value, because no exception will be thrown. If you ignore this code, your are not able to know if the DLL results are correct!

It is important to catch exceptions anyway, to be able to detect if the DLL could not be loaded and a Reframe object could not be instantiated, for example.

ComputeReframe

This method needs 7 arguments:

```
int ComputeReframe(ref double east, ref double north, ref double height,
                   int plaFrameIn, int plaFrameOut,
                   int altFrameIn, int altFrameOut)
```

[13] Topographic Landscape Model. Extent: emin=460’000 nmin=42’000 emax=868’000 nmax=322'000

1. **east:**
   Easting coordinate in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
2. **north:**
   Northing coordinate in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
3. **height:**
   Height in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output height after the processing.
4. **plaFrameIn:**
   The planimetric reference frame of your source data. This setting will be automatically checked when REFRAME looks for the corresponding coordinate in the triangular transformation network; if the input coordinates are outside of the definition bounds and so outside of the dataset extents, you will get an error. Available choices:
   - 0: Swiss plane coordinates LV03 (CH1903)
   - 1: Swiss plane coordinates LV95 (CH1903+)
5. **plaFrameOut:**
   The planimetric reference frame of your destination data. If you don't need a planimetric transformation but only want to compute an altimetric transformation, you have to use the same values for input and output. Available choices are same as for input:
   - 0: Swiss plane coordinates LV03 (CH1903)
   - 1: Swiss plane coordinates LV95 (CH1903+)
6. **altFrameIn:**
   The altimetric reference frame of your source data. This setting cannot be automatically defined or checked, so you have to be careful and ensure yourself that your choice really corresponds to the current source data. Otherwise you will get incorrect results and you won't notice anything! Available choices:
   - 0: National levelling network LN02 (levelled heights)
   - 1: National height network LHN95 (orthometric heights, CHGeo2004)
   - 2: Ellipsoidal heights (on Bessel 1841)
7. **altFrameOut:**
   The altimetric reference frame of your destination data. If you don't need an altimetric transformation but only want to compute a planimetric transformation, you have to use the same values for input and output. Available choices are same as for input:
   - 0: National levelling network (levelled heights)
   - 1: National height network LHN95 (orthometric heights, CHGeo2004)
   - 2: Ellipsoidal heights (on Bessel 1841)

This method returns an integer value, a code which defines if the transformation was done successfully, or describes the error if not:

- 1: Computation successfully executed: the coordinates passed by reference have been updated with the new output values
- -1: Error: specified point outside of the CHENyx06 triangular network (input coordinates outside boundaries)

  Note: for a planimetric transformation LV03↔LV95, if the input point is outside the CHENyx06 triangular network (what means outside the official Swiss TLM perimeter) the transformation is officially defined as a translation of +/- 2'000'000 m and +/- 1'000'000 m.
- -2: Error: specified point outside of the HTRANS or CHGeo2004 grid (input coordinates outside boundaries)

- -3: Error: problem occurred when reading a binary file. Check that all the binary files (datasets definitions) are correctly installed and valid (try to recover/recopy the original versions). Reinstall the REFRAME DLL if the problem persists.
- -4: Error: unsupported value for “plaFrameIn” or “plaFrameOut” argument (only “0” or “1” are allowed)
- -5: Error: unsupported value for “altFrameIn” or “altFrameOut” argument (only “0”, “1” or “2”[14] are allowed)
- -6: Error: input and output reference frames (planimetry and altimetry) are the same, there isn’t any transformation to do!
- -10: Error: CHENyx06 dataset inaccessible (from binary file “swisstopo.data.dll”). Check if the file exists in the application directory (next to “swisstopoReframeLib.dll”) and that it is accessible (enough rights). Reinstall the REFRAME DLL if the problem persists.
- -11: Error: HTRANS dataset inaccessible (from binary file “swisstopo.data.dll”). Check if the file exists in the application directory (next to “swisstopoReframeLib.dll”) and that it is accessible (enough rights). Reinstall the REFRAME DLL if the problem persists.
- -12 Error: CHGeo2004 dataset is inaccessible (from binary file “swisstopo.data.dll”). Check if the file exists in the application directory (next to “swisstopoReframeLib.dll”) and that it is accessible (enough rights). Reinstall the REFRAME DLL if the problem persists.

It is strongly recommended to check this value, because no exception will be thrown. If you ignore this code, your are not able to know if the DLL results are correct!

It is important to catch exceptions anyway, to be able to detect if the DLL could not be loaded and a Reframe object could not be instantiated, for example.

SetDatasetsDir

This method has been deprecated. It could be used to set the folder where the transformation datasets can be found. With the last release of the REFRAME DLL, all the dataset files have been embedded in one single binary file, which is automatically loaded from the same directory than the main DLL (“swisstopoReframeLib.dll”).

This method may only be used for test purposes, if you own the old distinct files (.bin and .grd).

It needs 1 argument:

```
void SetDatatsetsDir(string datasetsDirectory)
```

1. **datasetsDirectory:**
   The absolute path where the datasets binary files are installed (CHENyx06, HTRANS, CHGeo2004). You must always end your path with a directory separator character (“****”).

If you don’t call this method (which is the same as passing “**null**” as argument), the default location that will be expected is a **“datasets” subdirectory relative the DLL** (“swisstopoReframeLib.dll”).

If the path is not properly defined, you will get a “binary file not found” error message (return code -3, -10, -11 or -15) when trying to perform a computation (“ComputeReframe” method).

8.5 Code sample in native C++

For a use without COM, please refer to chapter 7.

Before you can use the DLL, you have to import the COM type library (*.tlb) “**swisstopoReframeLib.tlb**” and initialize the COM library with “**CoInitialize**”.

[14] “2” (= ellipsoidal heights) works only if the CHGeo2004 dataset is installed

At runtime, “swisstopoReframeLib.dll” will be found automatically due to the Windows registry entries make by the setup. The path to the type library as to be specified to the compiler. This can be done in the project or file (.cpp) properties, under “**Configuration Properties > C/C++ > General > Additional Include Directories**”.

Example of C++ code:

```
// Reference to REFRAME DLL
#import <swisstopoReframeLib.tlb>[15]
…

try
{
    // Initialize COM connection
    CoInitialize(NULL);

    // Create a pointer to the Reframe class from DLL
    swisstopoReframeLib::IReframePtr
        pReframe(__uuidof(swisstopoReframeLib::Reframe));

    // Input coordinates: read in a file, got from a textbox,
    // or obtained through another method or library...
    double c1 = 601000.0; // East in LV03
    double c2 = 197500.0; // North in LV03
    double c3 =    555.0; // Usual height (LN02)

    // Transform LV03 coord. to LV95 and LN02 height to Bessel
    int result = pReframe->ComputeReframe(&c1, &c2, &c3,
                                          0, 1, 0, 2);

    // Analyze result
    if (result==1) // OK
        … // TODO
    Else // Error
        … // TODO
}
catch (…)
{
    // unmanaged error, e.g. DLL cannot be loaded (COM error)
    … // TODO
}
```

After this code, c1 (east in LV95) = 2601000.030 m, c2 (north in LV95) = 1197500.037 m, c3 (ellipsoidal height on Bessel) = 554.335 m and result = 1 (if binary files are correctly installed).

[15] This will work if the REFRAME DLL and the type library are correctly registered in the system with “regasm” (refer to chapter 5.4). If you are experiencing problems when you try to instantiate a Reframe object or if you don’t want to use “/codebase” option with “regasm” tool, you can import the TLB file giving the full path (e.g. “C:\MyApplication\swisstopoReframeLib.tlb”). This will be less flexible if you deploy/redistribute your application.

```
// Succession of the first part above

try
{
    // Transform LV95 coordinates to ETRS89 longitude/latitude
    // and ellipsoidal height on Bessel to GRS80
    res = pReframe->ComputeGpsref(&c1, &c2, &c3, 3);

    // Analyze result
    if (result==1) // OK
        … // TODO
    Else // Error
        … // TODO
}
catch (…)
{
    // unmanaged error, e.g. DLL cannot be loaded (COM error)
    … // TODO
}
```

After this code, c1 (longitude) = 7.451764 °, c2 (latitude) = 46.928595 °, c3 (ellipsoidal height on GRS80) = 604.004 m and result = 1.

8.6 Code sample in Visual Basic or Visual Basic for Applications (VB/VBA)

Before you can use the component, you have to add a reference to **"swisstopo REFRAME library 2.0 (02.2012)" (swisstopoReframeLib.tlb)**.

Example of VB code:

```
'Exception handling
On Error GoTo ReframeError

'Create a new Reframe object
Dim oReframeLib As New swisstopoReframeLib.Reframe

'Input coordinates: read in a file, got from a textbox,
'or obtained through another method or library...
Dim c1 As Double, c2 As Double, c3 As Double
c1 = 601000# 'East in LV03
c2 = 197500# 'North in LV03
c3 = 555#    'Usual height (LN02)

'Transform LV03 coordinates to LV95 and LN02 height to Bessel
Dim intResult As Long
intResult = oReframeLib.ComputeReframe(c1, c2, c3, 0, 1, 0, 2)

'Analyze result
If intResult = 1 Then 'OK
    … 'TODO
Else 'Error
    … 'TODO
End If
```

After this code, c1 (east in LV95) = 2601000.030 m, c2 (north in LV95) = 1197500.037 m, c3 (ellipsoidal height on Bessel) = 554.335 m and result = 1 (if binary files correctly installed).

```
'Succession of the first part above

'Transform LV95 coordinates to ETRS89 longitude/latitude
'and ellipsoidal height on Bessel to GRS80
intResult = oReframeLib.ComputeGpsref(c1, c2, c3, 3)

'Analyze result
If intResult = 1 Then 'OK
    … 'TODO
Else 'Error
    … 'TODO
End If
```

After this code, c1 (longitude) = 7.451764 °, c2 (latitude) = 46.928595 °, c3 (ellipsoidal height on GRS80) = 604.004 m and result = 1.

Exception handling could be done at the end of the Function (or Sub) as following:

```
    On Error GoTo 0
    Exit Function

    'Reframe error / exception
ReframeError:
    … 'TODO
```

9 Using Silverlight DLL in Web applications

9.1 Main class

The REFRAME Silverlight DLL main class is named "**Reframe**" and is defined in the namespace "**swisstopo.geodesy.reframe.silverlight**".

The only one constructor takes no argument:

```
public Reframe()
```

9.2 Distinct types (enum)

- **PlanimetricFrame** (for use with "ComputeReframe" function):
  - **LV03_Military** = 0
    Swiss plane (military) coordinates LV03 (CH1903), EPSG code 21781
  - **LV95** = 1
    Swiss plane coordinates LV95 (CH1903+), EPSG code 2056
  - **LV03_Civil** = 2
    Swiss civil coordinates LV03C-G (CH1903), EPSG code 21782

- **AltimetricFrame** (for use with “ComputeReframe” function):
  - **LN02 = 0**
    National levelling network LN02 (levelled heights)
  - **LHN95 = 1**
    National height network LHN95 (orthometric heights on geoid “CHGeo2004”)
  - **Ellipsoid = 2**
    Ellipsoidal heights on Bessel 1841 (CH1903/CH1903+) or GRS80 (ETRS89)
  - **CHGeo98 = 3**
    Preliminary orthometric heights (on geoid “CHGeo98”)
- **ProjectionChange** (for use with “ComputeGpsref” function):
  - **ETRF93GeocentricToLV95** = 0
    Transformation of geocentric coordinates ETRF93 (ETRS89) [m] into Swiss plane coordinates LV95 (CH1903+) [m]
  - **ETRF93GeographicToLV95** = 1
    Transformation of geographic coordinates ETRF93 (ETRS89) [°] into Swiss plane coordinates LV95 (CH1903+) [m]
  - **LV95ToETRF93Geocentric** = 2
    Transformation of Swiss plane coordinates LV95 (CH1903+) [m] into geocentric coordinates ETRF93 (ETRS89) [m]
  - **LV95ToETRF93Geographic** = 3
    Transformation of Swiss plane coordinates LV95 (CH1903+) [m] into geographic coordinates ETRF93 (ETRS89) [m]

9.3 Methods and returns

- **ComputeGpsref**:
  Transformation of Swiss plane coordinates LV95 and ellipsoidal heights on Bessel to global geographic or geocentric CHTRS95/ETRS89/WGS84 coordinates and ellipsoidal heights on GRS80, or reverse.
- **ComputeReframe**:
  Transformation of Swiss plane coordinates LV03 to LV95 or reverse, and transformations between Swiss levelled heights LN02, orthometric heights LHN95 and ellipsoidal heights on Bessel.

ComputeGpsref

This method takes 4 arguments:

```
void ComputeGpsref(ref double east_x_lon, ref double north_y_lat,
                   ref double height_z,
                   ProjectionChange transformation);
```

1. **east_x_lon:**
   Swiss LV95 easting coordinate in meters [m], CHTRS95/ETRS89/WGS84 geocentric X coordinate in meters [m] or CHTRS95/ETRS89/WGS84 geographic longitude (λ) in decimal degrees [°]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
2. **north_y_lat:**
   Swiss LV95 northing coordinate in meters [m], CHTRS95/ETRS89/WGS84 geocentric Y coordinate in meters [m] or CHTRS95/ETRS89/WGS84 geographic latitude (φ) in decimal degrees [°]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.

3. **height_z:**
   Ellipsoidal Height in meters [m], on Bessel 1841 if input values are Swiss plane coordinates in LV95, on GRS80 if input values are global coordinates in CHTRS95/ETRS89/WGS84. This argument is passed by reference, which means that the input value is replaced by the output height after the processing.
4. **transformation:**
   A constant (distinct type) that defines the transformation to compute (input and output reference frames). Refer to paragraph 6.2, under "**ProjectionChange**", for possible values.

This method does not return any value.

It something fails, an exception is thrown. For example:

- ArgumentOutOfRangeException: Input coordinates are not LV95 (transformation from LV95 to ETRF93)
- Exception: General unmanaged exception. It should never happen, but it is strongly recommended to catch it to avoid unexpected results, for example because of an incompatible system.

The exception message gives more information about the error.

ComputeReframe

This method takes 7 arguments:

```
bool ComputeReframe(ref double east, ref double north, ref double height,
                    PlanimetricFrame plaFrameIn, PlanimetricFrame plaFrameOut,
                    AltimetricFrame altFrameIn, AltimetricFrame altFrameOut)
```

1. **east:**
   Easting coordinate in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
2. **north:**
   Northing coordinate in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output coordinate after the processing.
3. **height:**
   Height in meters [m]. This argument is passed by reference, which means that the input value is replaced by the output height after the processing.
4. **plaFrameIn:**
   The planimetric reference frame of your source data. Refer to paragraph 6.2, under "**PlanimetricFrame**", for possible values.
5. **plaFrameOut:**
   The planimetric reference frame of your destination data. Refer to paragraph 6.2, under "**PlanimetricFrame**", for possible values.
6. **altFrameIn:**
   The altimetric reference frame of your source data. Refer to paragraph 6.2, under "**AltimetricFrame**", for possible values.
7. **altFrameOut:**
   The altimetric reference frame of your destination data. Refer to paragraph 6.2, under "**AltimetricFrame**", for possible values.

This method returns a boolean value:

- **true**: Transformation successfully executed with the standard method (CHENyx06/FINELTRA) without any exception. Input point is in the official Swiss TLM[16] perimeter.
- **false**: Transformation performed with a constant shift of +/- 2'000'000.000 m east and +/- 1'000'000.000 m north, because the input coordinates are outside the official Swiss TLM perimeter.

It something fails, an exception is thrown. For example:

- ArgumentOutOfRangeException: Altimetry could not be transformed, because input coordinates are outside HTRANS or CHENyx06 official Swiss TLM perimeter.
- Exception: General unmanaged exception. It should never happen, but it is strongly recommended to catch it to avoid unexpected results, for example because of an incompatible system.

The exception message gives more information about the error.

Note: for a planimetric transformation LV03↔LV95, if the input point is outside the CHENyx06 triangular network (= outside the official swiss TLM perimeter), no exception is thrown because the transformation is officially defined as a translation of +/- 2'000'000 m and +/- 1'000'000 m. **You can get this information with the return value however**.

9.4 Code sample in C# (Silverlight Application)

Before you can use the DLL, you have to add a reference to "**swisstopo.reframeLibSl.dll**" in your Silverlight project. The namespace is "swisstopo.geodesy.reframe.silverlight".

Example of C# code:

```
// Reference to the REFRAME DLL
using swisstopo.geodesy.reframe.silverlight;
…

// Instantiate a Reframe object
Reframe reframeObj = new Reframe();

// Input coordinates: read in a file, got from a textbox,
// or obtained through another method...
double c1 = 601000.0; // East in LV03
double c2 = 197500.0; // North in LV03
double c3 =    555.0; // Usual height (LN02)

// Transform LV03 coordinates to LV95 and LN02 height to Bessel
try
{
    bool outsideChenyx06 =
          !reframeObj.ComputeReframe(ref c1, ref c2, ref c3,
                                     Reframe.PlanimetricFrame.LV03_Military,
                                     Reframe.PlanimetricFrame.LV95,
                                     Reframe.AltimetricFrame.LN02,
                                     Reframe.AltimetricFrame.Ellipsoid);
```

[16] Topographic Landscape Model. Extent: emin=460'000 nmin=42'000 emax=868'000 nmax=322'000

```
        // If outsideChenyx06==true a warning must perhaps be done!

        // c1, c2 and c3 now contain transformed coordinates...
        … // TODO
    }
    catch (ArgumentOutOfRangeException e)
    {
        // Coordinates outside Swiss TLM perimeter:
        // => height transformation impossible
        … // TODO
    }
    catch (Exception)
    {
        // other exception
        … // TODO
    }
```

After this code, c1 (east in LV95) = 2601000.030 m, c2 (north in LV95) = 1197500.037 m and c3 (ellipsoidal height on Bessel) = 554.335 m.

```
// Succession of the first part above

// Transform LV95 coordinates to ETRS89 longitude/latitude
// and ellipsoidal height on Bessel to GRS80
try
{
    reframeObj.ComputeGpsref(ref c1, ref c2, ref c3,
                Reframe.ProjectionChange.LV95ToETRF93Geographic);

    // c1, c2 and c3 now contain transformed coordinates...
    … // TODO
}
catch (Exception)
{
    // exception
}
```

After this code, c1 (longitude) = 7.451764 °, c2 (latitude) = 46.928595 ° and c3 (ellipsoidal height on GRS80) = 604.004 m.

10 Using the Java classes in desktop or Web applications

10.1 Main class

The REFRAME Java library main class is named “**Reframe**” and is accessible through an interface named “**IReframe**”. It is defined in the package “**com.swisstopo.geodesy.reframe_lib**”.

The only one constructor takes no argument:

```
public Reframe(String datasetsDirectory)
```

10.2 Distinct types (enum)

- **PlanimetricFrame** (for use with “ComputeReframe” function):
  - **LV03_Military**
    Swiss plane (military) coordinates LV03 (CH1903), EPSG code 21781
  - **LV95**
    Swiss plane coordinates LV95 (CH1903+), EPSG code 2056
  - **LV03_Civil**
    Swiss civil coordinates LV03C-G (CH1903), EPSG code 21782
- **AltimetricFrame** (for use with “ComputeReframe” function):
  - **LN02**
    National levelling network LN02 (levelled heights)
  - **LHN95**
    National height network LHN95 (orthometric heights on geoid “CHGeo2004”)
  - **Ellipsoid**
    Ellipsoidal heights on Bessel 1841 (CH1903/CH1903+) or GRS80 (ETRS89)
- **ProjectionChange** (for use with “ComputeGpsref” function):
  - **ETRF93GeocentricToLV95**
    Transformation of geocentric coordinates ETRF93 (ETRS89) [m] into Swiss plane coordinates LV95 (CH1903+) [m]
  - **ETRF93GeographicToLV95**
    Transformation of geographic coordinates ETRF93 (ETRS89) [°] into Swiss plane coordinates LV95 (CH1903+) [m]
  - **LV95ToETRF93Geocentric**
    Transformation of Swiss plane coordinates LV95 (CH1903+) [m] into geocentric coordinates ETRF93 (ETRS89) [m]
  - **LV95ToETRF93Geographic**
    Transformation of Swiss plane coordinates LV95 (CH1903+) [m] into geographic coordinates ETRF93 (ETRS89) [m]

10.3 Methods and returns

- **ComputeGpsref**:
  Transformation of Swiss plane coordinates LV95 and ellipsoidal heights on Bessel to global geographic or geocentric CHTRS95/ETRS89/WGS84 coordinates and ellipsoidal heights on GRS80, or reverse.
- **ComputeReframe**:
  Transformation of Swiss plane coordinates LV03 to LV95 or reverse, and transformations between Swiss levelled heights LN02, orthometric heights LHN95 and ellipsoidal heights on Bessel.

ComputeGpsref

This method takes 2 arguments:

```
public double[] ComputeGpsref(double[] coordinates,
                  ProjectionChange transformation)
                  throws IllegalArgumentException
```

1. **coordinates:**
   Swiss LV95 coordinates (easting and northing) and ellipsoidal height in meters [m], CHTRS95/ETRS89/WGS84 geocentric X/Y/Z coordinates in meters [m] or CHTRS95/ETRS89/WGS84 geographic longitude (λ) and latitude (φ) in decimal degrees [°] and altitude in meters [m].
2. **transformation:**
   A constant (distinct type) that defines the transformation to compute (input and output reference frames). Refer to paragraph 0, under "**ProjectionChange**", for possible values.

This method returns an array of doubles:

- **double[]**: Three-dimensional array of doubles containing the result coordinates (E/N/H in [m], X/Y/Z in [m] or λ/φ in [°] and h in [m]).

It something fails, an exception is thrown. For example:

- IllegalArgumentException: Input coordinates are not LV95 (transformation from LV95 to ETRF93)
- Exception: General unmanaged exception. It should never happen, but it is strongly recommended to catch it to avoid unexpected results, for example because of an incompatible system.

The exception message gives more information about the error.

ComputeReframe

This method takes 5 arguments:

```
public double[] ComputeReframe(double[] coordinates,
                PlanimetricFrame plaFrameIn, PlanimetricFrame plaFrameOut,
                AltimetricFrame altFrameIn, AltimetricFrame altFrameOut)
                throws IllegalArgumentException
```

1. **coordinates:**
   Point coordinates (easting, northing and height) in meters [m].
2. **plaFrameIn:**
   The planimetric reference frame of your source data. Refer to paragraph 0, under "**PlanimetricFrame**", for possible values.
3. **plaFrameOut:**
   The planimetric reference frame of your destination data. Refer to paragraph 0, under "**PlanimetricFrame**", for possible values.
4. **altFrameIn:**
   The altimetric reference frame of your source data. Refer to paragraph 0, under "**AltimetricFrame**", for possible values.
5. **altFrameOut:**
   The altimetric reference frame of your destination data. Refer to paragraph 0, under "**AltimetricFrame**", for possible values.

This method returns an array of doubles:

- **double[]**: Three-dimensional array of doubles containing the result coordinates (E/N/H in [m]).

It something fails, an exception is thrown. For example:

- IllegalArgumentException: Altimetry could not be transformed, because input coordinates are outside HTRANS official Swiss TLM perimeter.
- NullPointerException: Transformation dataset files resource could not be loaded. Check if the reframeLib.jar file is correctly linked and accessible.
- Exception: General unmanaged exception. It should never happen, but it is strongly recommended to catch it to avoid unexpected results, for example because of an incompatible system.

The exception message gives more information about the error.

Note: for a planimetric transformation LV03↔LV95, if the input point is outside the CHENyx06 triangular network (= outside the official Swiss TLM perimeter), no exception is thrown because the transformation is officially defined as a translation of +/- 2'000'000 m and +/- 1'000'000 m.

10.4 Code sample in Java

Before you can use the class, you have to link the JAR file in your project and import the package "**com.swisstopo.geodesy.reframe_lib.***".

Example of Java code:

```
…
// Import the REFRAME package
import com.swisstopo.geodesy.reframe_lib.*;
import com.swisstopo.geodesy.reframe_lib.IReframe.AltimetricFrame;
import com.swisstopo.geodesy.reframe_lib.IReframe.PlanimetricFrame;
import com.swisstopo.geodesy.reframe_lib.IReframe.ProjectionChange;
…

// Instantiate a Reframe object
Reframe reframeObj = new Reframe();

// Input coordinates: read in a file, got from a text field,
// or obtained through another method or library...
// East and North in LV03, usual height (LN02)
double[] inputCoordinates = new double[] { 601000.0, 197500.0, 555.0 };

// Transform LV03 coordinates to LV95 and LN02 height to Bessel
try
{
     double[] outputCoordinates =
           !reframeObj.ComputeReframe(inputCoordinates,
                                      PlanimetricFrame.LV03_Military,
                                      PlanimetricFrame.LV95,
                                      AltimetricFrame.LN02,
                                      AltimetricFrame.Ellipsoid);

     // outputCoordinates now contains transformed coordinates...
     … // TODO
 }
```

```
catch (IllegalArgumentException e)
{
    // Coordinates outside Swiss TLM perimeter:
    // => height transformation impossible
    … // TODO
}
catch (NullPointerException e)
{
    // A binary dataset file was not correctly loaded
    … // TODO
}
catch (Exception)
{
    // other exception
    … // TODO
}
```

After this code, outputCoordinates[0] (east in LV95) = 2601000.030 m, outputCoordinates[1] (north in LV95) = 1197500.037 m and outputCoordinates[2] (ellipsoidal height on Bessel) = 554.335 m.

```
// Succession of the first part above

// Transform LV95 coordinates to ETRS89 longitude/latitude
// and ellipsoidal height on Bessel to GRS80
try
{
    outputCoordinates = reframeObj.ComputeGpsref(outputCoordinates,
                            ProjectionChange.LV95ToETRF93Geographic);

    // outputCoordinates now contains transformed coordinates...
    … // TODO
}
catch (Exception)
{
    // exception
}
```

After this code, outputCoordinates[0] (longitude) = 7.451764 °, outputCoordinates[1] (latitude) = 46.928595 ° and outputCoordinates[2] (ellipsoidal height on GRS80) = 604.004 m.

Wabern, April 2016

Federal Office of Topography swisstopo
Geodetic developments and contracts

Jérôme Ray