



APSALIN COMPANY

ApsNadcon™ .NET DLL, Version 2.0.0

ApsNadcon™ User's Guide

© APSalin Company
All rights reserved.

www.apsalin.com

TRADEMARKS

APSalin, APSalin Company, ApsNadcon, ApsNadcon DLL, the APSalin Company logo, the ApsNadcon logo, the ApsNadcon DLL logo are trademarks of APSalin Company.

.NET™, and Visual Studio.NET™ are trademarks of Microsoft, Inc.

All other trademarks are property of their respective owners.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THIS PUBLICATION COULD CONTAIN TYPOGRAPHIC ERRORS AND/OR TECHNICAL INACCURACIES. UPDATES AND MODIFICATIONS MAY BE MADE TO THIS DOCUMENT AND/OR SUPPORTING SOFTWARE AT ANY TIME.

APSalin Company has intellectual property rights relating to technology embodied in this product. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and/or other countries.

This product is distributed under licenses restricting its use, copying, distribution, and de-compilation. No part of this product may be reproduced in any form by any means without prior written authorization of APSalin Company.

Table of Contents

Introduction	1
Installation.....	2
Licensing.....	6
Programming Reference	7
Overview	7
ApsNadconRegion Enumeration.....	10
ApsNadcon.ConvertToNad27 Functions.....	10
ApsNadcon.ConvertToNad83 Functions.....	12
ApsNadcon.FindNadconRegion Function	13
ApsNadcon.IsInsideNadconBoundaries Function	14
ApsNadcon.IsInsideNadconRegion Function.....	14
ApsNadcon.IsValidLatitude Function	15
ApsNadcon.IsValidLongitude Function	15
ApsNadcon.IsValidNadconRegion Function	15
C# Examples.....	17



Introduction

NADCON stands for North American Datum Conversion. The original NADCON method was developed by National Geodetic Survey (NGS) and was adopted by the Federal Geodetic Control Committee (FGCC) as a standard method for mathematical transformations between the North American Datum of 1927 (NAD 27) and the North American Datum of 1983 (NAD 83) on August 10th, 1990.

ApsNadcon™ is .NET Framework assembly / library that provides geographic coordinate conversion / transformation between the North American Datum of 1927 (NAD 27) and the North American Datum of 1983 (NAD 83) within the U.S. territory based on the original NGS NADCON method.

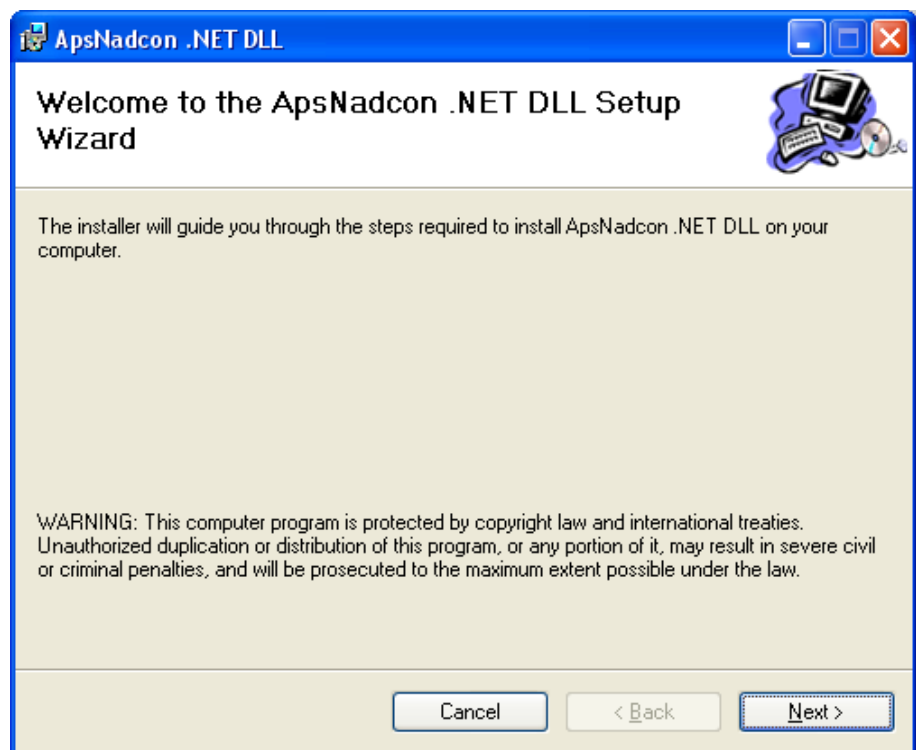
ApsNadcon™ library:

- Includes conversion functions that are based on the original NGS method.
- Generates conversion results that are identical to NGS NADCON results.
- Includes functions to check if a geographic coordinate is inside NADCON or the particular NADCON region boundaries.
- Contains single DLL file.
- Is ready for Microsoft .NET applications (C#, C++, Visual Basic).

Installation

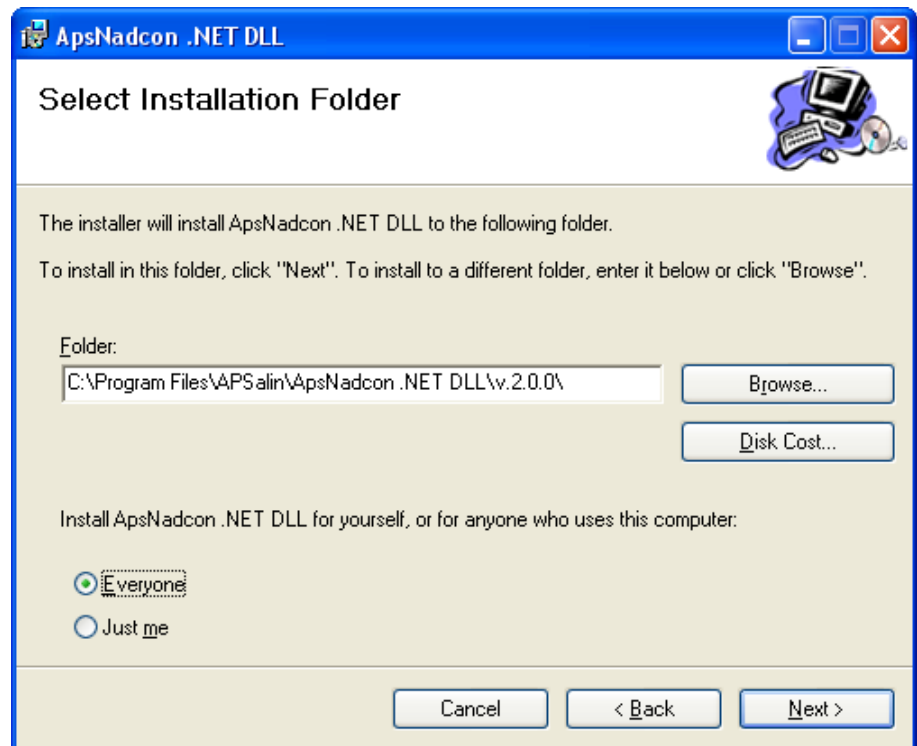
To install ApsNadcon™:

1. Run **ApsNadcon.msi** file
2. Press Next



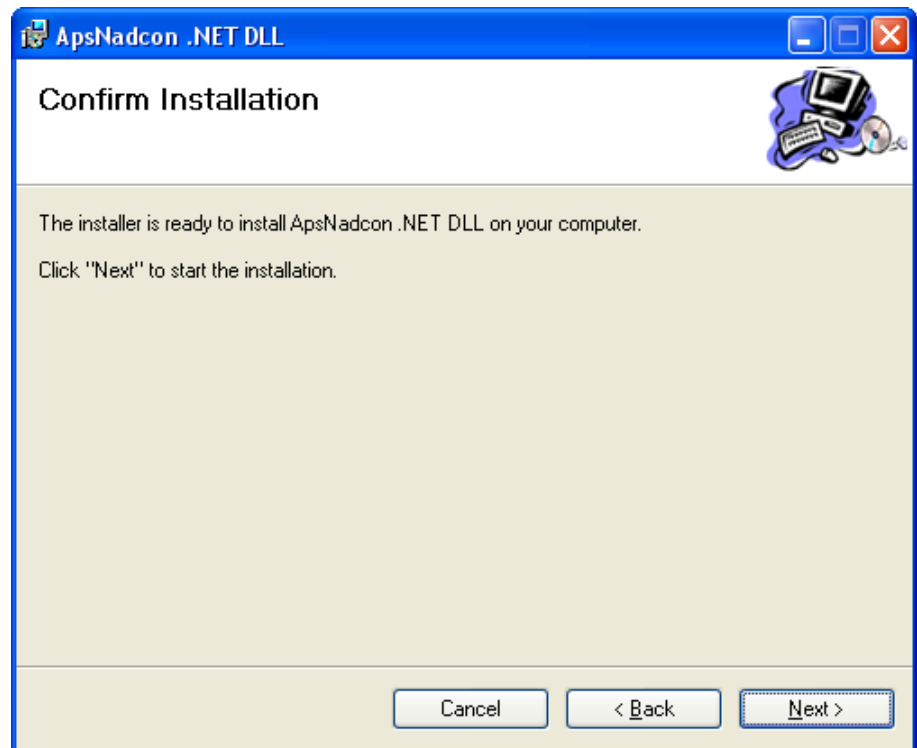
INSTALLATION

3. Select installation folder and installation scope, then press Next



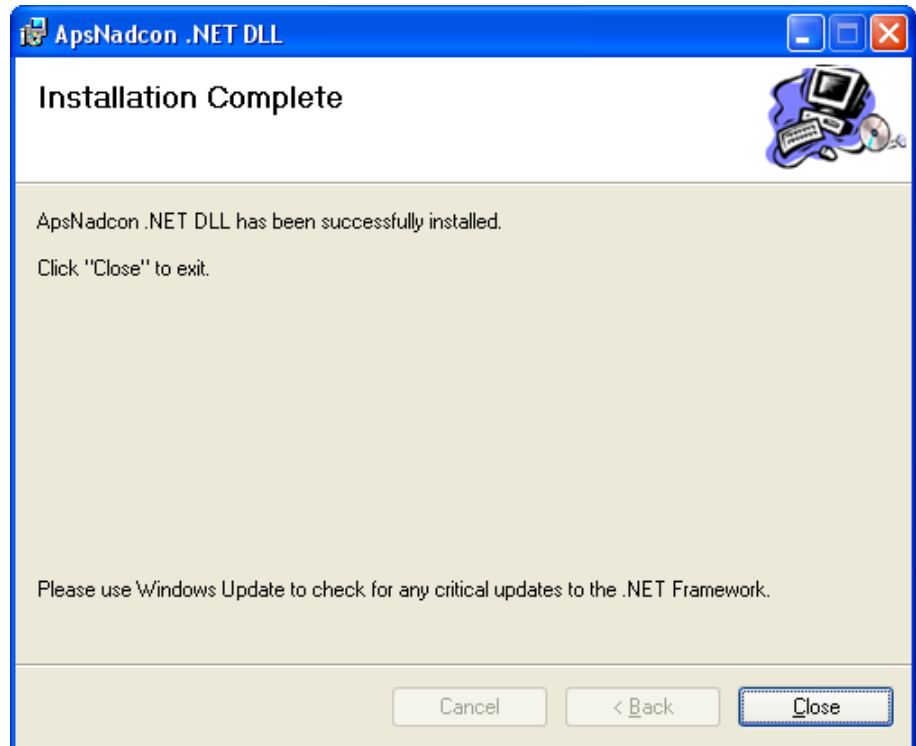
INSTALLATION

4. Select installation folder and installation scope, then press Next



INSTALLATION

5. Wait until installation is complete and then press Close.



6. Navigate to the folder which you entered earlier, it should contain the following files:
 - ApsNadcon.dll – the library itself. You either copy it to the desired new location or set a reference from your project on this location. If you copy it a new location, then you have to copy the valid end user license file (ApsNadcon.license) along with it.
 - ApsNadcon.XML – help file, which contains information used by IntelliSense in Microsoft Visual Studio. It should be in the same directory with the ApsNadcon.DLL file.
 - ApsNadconNetDllV2License.pdf – the text of the end user license.
 - ApsNadconNetDllV2UserGuide.pdf – user's guide.
 - ApsNadcon.license – the valid end user license. It should be always present in the same directory with the ApsNadcon.DLL file.
 - The Help folder that contains HTML and MSDN style documentation.

Licensing

Below are the extracts from ApsNadcon™ end user license, the full version can be found under <INSTALLATION PATH>\License.txt

- General Software License Grant. You are granted a right to install the Software Product and use it in order to process your own software with the Software Product. APSalin grants the use of the Software Product according to one of the license types below as identified in the product title. Licenses purchased for development do not extend to third parties even if the software was developed for that third party. All entities must have a separate license.
- Single Developer License. If you purchased a Single Developer License, APSalin grants to you one (1) personal, nontransferable, nonexclusive, royalty-free license to use copies of the Software Product and install such Software on your machine(s) for your single concurrent internal use. Use of the Software Product by other individuals is permitted only if said other individual has been licensed to use the Software Product.
- Redistribution. You are granted a royalty-free license to redistribute in binary form any components of the Software Product explicitly marked as redistributable provided that (a) you provide all technical support for the distribution, (b) you include APSalin copyright notice for the Software Product on the title page of the documentation for your software application product; and (c) you do not allow recipients to disassemble, decompile, or in any other way allowing them to gain separate access to the Software Product or any part of the Software Product, and (d) you agree to indemnify, hold harmless and defend APSalin from and against and pay any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of your software application product, and (e) APSalin is not obligated to provide support for works derived from the Software Product.
- No Refund. Once you pay the license fee, your payment is final and you may not be refunded.

Programming Reference

Overview

ApsNadcon™ .NET Framework assembly / library provides geographic coordinate conversion / transformation between the North American Datum of 1927 (NAD 27) and the North American Datum of 1983 (NAD 83) within the U.S. territory based on the original NGS NADCON method.

ApsNadcon™ uses seven (7) regions / files of gridded datum shift data provided by NGS and employs bilinear interpolation to determine latitude / longitude correction.

ApsNadcon™ contains the following items:

- a. ApsNadconRegion enumeration – list of all NADCON regions. For details see TABLE 4.1 below.
- b. ApsNadcon class – contains all conversion related functions. For details see TABLE 4.2 below.

PROGRAMMING REFERENCE

Table 4.1: list of NADCON regions

Region	Description	Min Lat	Max Lat	Min Lon	Max Lon
ConterminousUS	Conterminous U.S. (lower 48 states)	20	50	-131	-63
HawaiiinIslands	Hawaii Islands	18	23	-161	-154
PuertoRicoVirginIslands	Puerto Rico and the Virgin Islands	17	19	-68	-64
StLawrenceIsland	St. Lawrence Is., AK – old datum within Alaska	62	64	-172	-168
StGeorgeIsland	St. George Is., AK – old datum within Alaska	56	57	-171	-169
StPaulIsland	St. Paul Is., AK – old datum within Alaska	57	58	-171	-169
Alaska	Alaska, including Aleutian Islands	46	77	-194	-128

PROGRAMMING REFERENCE

**Table 4.2:
functions**

Function	Description
<code>ApsNadcon.ConvertToNad27</code>	Converts a geographic coordinate to NAD 27 datum.
<code>ApsNadcon.ConvertToNad83</code>	Converts a geographic coordinate to NAD 83 datum.
<code>ApsNadcon.FindNadconRegion</code>	Finds NADCON region.
<code>ApsNadcon.IsInsideNadconBoundaries</code>	Checks if the specified geographic coordinate is inside at least one of the NADCON regions.
<code>ApsNadcon.IsInsideNadconRegion</code>	Checks if a geographic coordinate is inside the specified NADCON region.
<code>ApsNadcon.IsValidLatitude</code>	Checks validity of a latitude.
<code>ApsNadcon.IsValidLongitude</code>	Checks validity of a longitude.
<code>ApsNadcon.IsValidNadconRegion</code>	Check validity of a NADCON region.

ApsNadconRegion Enumeration

This enumeration represents the list of seven (7) NADCON regions. For details see TABLE 4.1.

```
public enum ApsNadconRegion : byte {
    /// <summary>
    /// Not defined
    /// </summary>
    NotDefined = 0,
    /// <summary>
    /// Conterminous U.S. (lower 48 states)
    /// </summary>
    ConterminousUS = 1,
    /// <summary>
    /// Hawaii Islands
    /// </summary>
    HawaiiIslands = 2,
    /// <summary>
    /// Puerto Rico and the Virgin Islands
    /// </summary>
    PuertoRicoVirginIslands = 3,
    /// <summary>
    /// St. Lawrence Is., AK - old datum within Alaska
    /// </summary>
    StLawrenceIsland = 4,
    /// <summary>
    /// St. George Is., AK - old datum within Alaska
    /// </summary>
    StGeorgeIsland = 5,
    /// <summary>
    /// St. Paul Is., AK - old datum within Alaska
    /// </summary>
    StPaulIsland = 6,
    /// <summary>
    /// Alaska, including Aleutian Islands
    /// </summary>
    Alaska = 7
}
```

ApsNadcon.ConvertToNad27 Functions

These functions convert the specified geographic coordinate in NAD83 to NAD27 datum. There are two (2) functions.

The first function:

```
public static bool ConvertToNad27(
```

PROGRAMMING REFERENCE

```
double latInDegNad83,  
double lonInDegNad83,  
out double latInDegNad27,  
out double lonInDegNad27,  
out double latShiftInSec,  
out double lonShiftInSec,  
out string errorMessage  
)
```

This function converts a geographic coordinate in NAD83 to NAD27 datum in the specified NADCON region.

It returns true - if the conversion succeeded, false - if it failed.

The function verifies validity of its arguments before running the conversion. The valid latitude is a real number in the range from -90.0 (inclusive) to 90.0 (inclusive). The valid longitude is a real number in the range from -180.0 (inclusive) to 180.0 (inclusive).

The function returns false, if the original geographic coordinate is outside NADCON boundaries.

The second function:

```
public static bool ConvertToNad27(  
    double latInDegNad83,  
    double lonInDegNad83,  
    ApsNadconRegion nadconRegion,  
    out double latInDegNad27,  
    out double lonInDegNad27,  
    out double latShiftInSec,  
    out double lonShiftInSec,  
    out string errorMessage  
)
```

This function converts a geographic coordinate in NAD83 to NAD27 datum in the specified NADCON region.

It returns true - if the conversion succeeded, false - if it failed.

The function verifies validity of its arguments before running the conversion. The valid latitude is a real number in the range from -90.0 (inclusive) to 90.0 (inclusive). The valid longitude is a real number in the range from -180.0 (inclusive) to 180.0 (inclusive).

The function returns false, if the original geographic coordinate is outside the specified NADCON region.

ApsNadcon.ConvertToNad83 Functions

These functions convert the specified geographic coordinate in NAD27 to NAD83 datum. There are two (2) functions.

The first function:

```
public static bool ConvertToNad83(  
    double latInDegNad27,  
    double lonInDegNad27,  
    out double latInDegNad83,  
    out double lonInDegNad83,  
    out double latShiftInSec,  
    out double lonShiftInSec,  
    out string errorMessage  
)
```

This function converts a geographic coordinate in NAD27 to NAD83 datum in the specified NADCON region.

It returns true - if the conversion succeeded, false - if it failed.

The function verifies validity of its arguments before running the conversion. The valid latitude is a real number in the range from -90.0 (inclusive) to 90.0 (inclusive). The valid longitude is a real number in the range from -180.0 (inclusive) to 180.0 (inclusive).

The function returns false, if the original geographic coordinate is outside NADCON boundaries.

The second function:

```
public static bool ConvertToNad83(  
    double latInDegNad27,  
    double lonInDegNad27,  
    ApsNadconRegion nadconRegion,  
    out double latInDegNad83,  
    out double lonInDegNad83,  
    out double latShiftInSec,  
    out double lonShiftInSec,  
    out string errorMessage  
)
```

This function converts a geographic coordinate in NAD27 to NAD83 datum in the specified NADCON region.

It returns true - if the conversion succeeded, false - if it failed.

The function verifies validity of its arguments before running the conversion. The valid latitude is a real number in the range from -90.0 (inclusive) to 90.0 (inclusive). The valid longitude is a real number in the range from -180.0 (inclusive) to 180.0 (inclusive).

The function returns false, if the original geographic coordinate is outside the specified NADCON region.

ApsNadcon.FindNadconRegion Function

This function determines NADCON region to which a geographic coordinate belongs.

```
public static bool FindNadconRegion(  
    double latInDeg,  
    double lonInDeg,  
    ApsNadconRegion nadconRegion,  
    out string errorMessage
```

)

It returns true if the NADCON region is found and false if it is not found or an error occurred.

The function verifies validity of its arguments before running the conversion. The valid latitude is a real number in the range from -90.0 (inclusive) to 90.0 (inclusive). The valid longitude is a real number in the range from -180.0 (inclusive) to 180.0 (inclusive).

ApsNadcon.IsInsideNadconBoundaries Function

This function checks if a geographic coordinate is inside at least one of the NADCON regions.

```
public static bool IsInsideNadconRegion(  
    double latInDeg,  
    double lonInDeg,  
    out string errorMessage  
)
```

It returns true if the specified geographic coordinate is inside NADCON boundaries, and false if it is outside or an error occurred.

The function verifies validity of its arguments before running the conversion. The valid latitude is a real number in the range from -90.0 (inclusive) to 90.0 (inclusive). The valid longitude is a real number in the range from -180.0 (inclusive) to 180.0 (inclusive).

ApsNadcon.IsInsideNadconRegion Function

This function checks if a geographic coordinate is inside the specified NADCON region.

```
public static bool IsInsideNadconRegion(  
    double latInDeg,  
    double lonInDeg,  
    ApsNadconRegion nadconRegion,  
    out string errorMessage  
)
```

It returns true if the specified geographic coordinate is inside the specified NADCON region, and false if it is outside or an error occurred.

The function verifies validity of its arguments before running the conversion. The valid latitude is a real number in the range from -90.0 (inclusive) to 90.0 (inclusive). The valid longitude is a real number in the range from -180.0 (inclusive) to 180.0 (inclusive).

ApsNadcon.IsValidLatitude Function

This function checks if the provided latitude (in decimal degrees) is valid.

```
public static bool IsValidLatitude(  
    double latInDeg,  
    out string errorMessage  
)
```

It returns true if the latitude is valid, false - if it is invalid.

The valid latitude is a real number in the range from -90.0 (inclusive) to 90.0 (inclusive).

ApsNadcon.IsValidLongitude Function

This function checks if the provided longitude (in decimal degrees) is valid.

```
public static bool IsValidLongitude(  
    double lonInDeg,  
    out string errorMessage  
)
```

It returns true if the longitude is valid, false - if it is invalid.

The valid longitude is a real number in the range from -180.0 (inclusive) to 180.0 (inclusive).

ApsNadcon.IsValidNadconRegion Function

This function checks if the provided NADCON region is valid, i.e. it is not undefined and it is not outside its range.

```
public static bool IsValidNadconRegion(  

```

PROGRAMMING REFERENCE

```
    ApsNadconRegion nadconRegion,  
    out string errorMessage  
)
```

It returns true if the NADCON region is valid, false - if it is invalid.

C# Examples

The installation contains C# example project, it can be found under the <INSTALLATION PATH>\Examples\C#\ApsNadconExampleCSharp directory.

In your projects remember to add reference on ApsNadcon™ library and the following line:

```
using ApsNadcon;
```

Below is the C# code that lists several scenarios of NADCON conversions with ApsNadcon™ library.

**Example 1:
conversion from
NAD83 to NAD27,
for specific
NADCON region**

```
double latInDegNad83 = 45.13;
double lonInDegNad83 = -120.1313;

ApsNadconRegion nadconRegion = ApsNadconRegion.ConterminousUS;

double latInDegNad27 = Double.NaN;
double lonInDegNad27 = Double.NaN;
double latShiftInSec = Double.NaN;
double lonShiftInSec = Double.NaN;

string errorMessage = String.Empty;

Console.WriteLine("Lat NAD83: " + latInDegNad83.ToString());
Console.WriteLine("Lon NAD83: " + lonInDegNad83.ToString());

if (!ApsNadcon.ConvertToNad27(
    latInDegNad83,
    lonInDegNad83,
    nadconRegion,
    out latInDegNad27,
    out lonInDegNad27,
    out latShiftInSec,
    out lonShiftInSec,
    out errorMessage))
{
    Console.WriteLine("Error: " + errorMessage);
}
```

C# EXAMPLES

```
}
else
{
    Console.WriteLine(
        "Lat NAD27: " + latInDegNad27.ToString());
    Console.WriteLine(
        "Lon NAD27: " + lonInDegNad27.ToString());
}
```

Example 2: conversion from NAD83 to NAD27

```
double latInDegNad83 = 45.13;
double lonInDegNad83 = -120.1313;

double latInDegNad27 = Double.NaN;
double lonInDegNad27 = Double.NaN;
double latShiftInSec = Double.NaN;
double lonShiftInSec = Double.NaN;

string errorMessage = String.Empty;

Console.WriteLine("Lat NAD83: " + latInDegNad83.ToString());
Console.WriteLine("Lon NAD83: " + lonInDegNad83.ToString());

if (!ApsNadcon.ConvertToNad27(latInDegNad83, lonInDegNad83, out
    latInDegNad27, out lonInDegNad27, out latShiftInSec, out
    lonShiftInSec, out errorMessage))
{
    Console.WriteLine("Error: " + errorMessage);
}
else
{
    Console.WriteLine("Lat NAD27: " + latInDegNad27.ToString());
    Console.WriteLine("Lon NAD27: " + lonInDegNad27.ToString());
}
```

Example 3: conversion from NAD27 to NAD83, for specific NADCON region

```
double latInDegNad27 = 45.13;
double lonInDegNad27 = -120.1313;

ApsNadconRegion nadconRegion = ApsNadconRegion.ConterminousUS;

double latInDegNad83 = Double.NaN;
double lonInDegNad83 = Double.NaN;
double latShiftInSec = Double.NaN;
double lonShiftInSec = Double.NaN;

string errorMessage = String.Empty;

Console.WriteLine("Lat NAD27: " + latInDegNad27.ToString());
Console.WriteLine("Lon NAD27: " + lonInDegNad27.ToString());
```

C# EXAMPLES

```
if (!ApsNadcon.ConvertToNad83(latInDegNad27, lonInDegNad27,
nadconRegion, out latInDegNad83, out lonInDegNad83, out
latShiftInSec, out lonShiftInSec, out errorMessage))
{
    Console.WriteLine("Error: " + errorMessage);
}
else
{
    Console.WriteLine("Lat NAD83: " + latInDegNad83.ToString());
    Console.WriteLine("Lon NAD83: " + lonInDegNad83.ToString());
}
```

Example 4: conversion from NAD27 to NAD83

```
double latInDegNad27 = 45.13;
double lonInDegNad27 = -120.1313;

double latInDegNad83 = Double.NaN;
double lonInDegNad83 = Double.NaN;
double latShiftInSec = Double.NaN;
double lonShiftInSec = Double.NaN;

string errorMessage = String.Empty;

Console.WriteLine("Lat NAD27: " + latInDegNad27.ToString());
Console.WriteLine("Lon NAD27: " + lonInDegNad27.ToString());

if (!ApsNadcon.ConvertToNad83(latInDegNad27, lonInDegNad27, out
latInDegNad83, out lonInDegNad83, out latShiftInSec, out
lonShiftInSec, out errorMessage))
{
    Console.WriteLine("Error: " + errorMessage);
}
else
{
    Console.WriteLine("Lat NAD83: " + latInDegNad83.ToString());
    Console.WriteLine("Lon NAD83: " + lonInDegNad83.ToString());
}
```

Example 5: finding NADCON region

```
double latInDeg = 45.13;
double lonInDeg = -120.1313;

ApsNadconRegion nadconRegion = ApsNadconRegion.ConterminousUS;

string errorMessage = String.Empty;

Console.WriteLine("Lat: " + latInDeg.ToString());
Console.WriteLine("Lon: " + lonInDeg.ToString());
```

C# EXAMPLES

```
if (!ApsNadcon.FindNadconRegion(latInDeg, lonInDeg, out
nadconRegion, out errorMessage))
{
    Console.WriteLine("Not found: " + errorMessage);
}
else
{
    Console.WriteLine("Found:" + nadconRegion);
}
```

Example 6: is inside NADCON boundaries

```
double latInDeg = 45.13;
double lonInDeg = -120.1313;

string errorMessage = String.Empty;

Console.WriteLine("Lat: " + latInDeg.ToString());
Console.WriteLine("Lon: " + lonInDeg.ToString());

if (!ApsNadcon.IsInsideNadconBoundaries(latInDeg, lonInDeg, out
errorMessage))
{
    Console.WriteLine("Is outside or error: " + errorMessage);
}
else
{
    Console.WriteLine("Inside");
}
```

Example 7: is inside NADCON region

```
double latInDeg = 45.13;
double lonInDeg = -120.1313;

ApsNadconRegion nadconRegion = ApsNadconRegion.ConterminousUS;

string errorMessage = String.Empty;

Console.WriteLine("Lat: " + latInDeg.ToString());
Console.WriteLine("Lon: " + lonInDeg.ToString());

if (!ApsNadcon.IsInsideNadconRegion(latInDeg, lonInDeg,
nadconRegion, out errorMessage))
{
    Console.WriteLine("Is outside or error: " + errorMessage);
}
else
{
    Console.WriteLine("Inside");
}
```

C# EXAMPLES

Example 8: is valid latitude

```
double latInDeg = 45.13;
string errorMessage = String.Empty;

if (ApsNadcon.IsValidLatitude(latInDeg, out errorMessage))
{
    Console.WriteLine("Latitude is valid.");
}
else
{
    Console.WriteLine(errorMessage);
}
```

Example 9: is valid longitude

```
double lonInDeg = -120.1313;
string errorMessage = String.Empty;

if (ApsNadcon.IsValidLongitude(lonInDeg, out errorMessage))
{
    Console.WriteLine("Longitude is valid.");
}
else
{
    Console.WriteLine(errorMessage);
}
```

Example 10: is valid NADCON region

```
ApsNadconRegion nadconRegion = ApsNadconRegion.ConterminousUS;
string errorMessage = String.Empty;

if (ApsNadcon.IsValidNadconRegion(nadconRegion, out errorMessage))
{
    Console.WriteLine("NADCON region is valid.");
}
else
{
    Console.WriteLine(errorMessage);
}
```