Autodesk University 2012

From Mobile and Through the Cloud to AutoCAD® Civil 3D®

Augusto Goncalves - Autodesk

CP3325 This class will show you how to collect coordinate points using mobile with GPS-enabled devices running Android™ or Apple® iOS, and then how to store and merge those in the cloud to finally generate AutoCAD Civil 3D objects, like surfaces, alignments, or parcels. Prior knowledge of .NET programming and the AutoCAD Civil 3D API are required.

Learning Objectives

At the end of this class, you will be able to:

- Understand cloud development with mobile and desktop
- Create, set-up and deploy a website on Azure cloud service
- Configure, develop and install Android application
- Connect Civil 3D to webservice (cloud) with .NET API

About the Speaker

Augusto is member of Autodesk DevTech team since 2008 based at Sao Paulo office. He is Civil Engineer with a Master in Computer Science, also a specialist in AutoCAD, Civil3D, Revit and Inventor APIs.
augusto.goncalves@autodesk.com

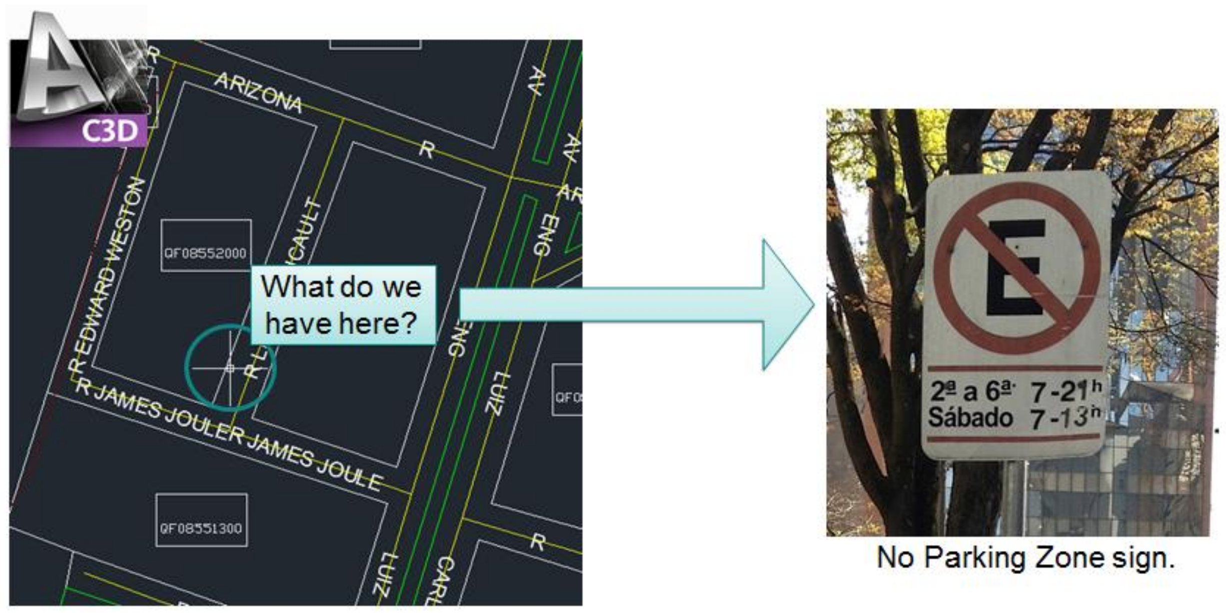
Overview

This section will present a general overview with the requirements, based on the flow of information, the architecture selected and the technologies used on the implementation.

Flow of Information

On a daily basis, an engineer often needs to collect data from the field and place into a Civil 3D project. There are some challenges to accomplish this: first it is required to collect the data on the field maintaining its correlation with the project, in other words, geo-referenced data. Second, it's required to include the data on the project.

For a given Civil 3D project that represents the actual site, at a specific point, the engineer may need to know if is there anything there, for instance, a 'No Parking' sign. The image below show this flow, the sign is in Portuguese, right across Autodesk office at São Paulo, Brazil.

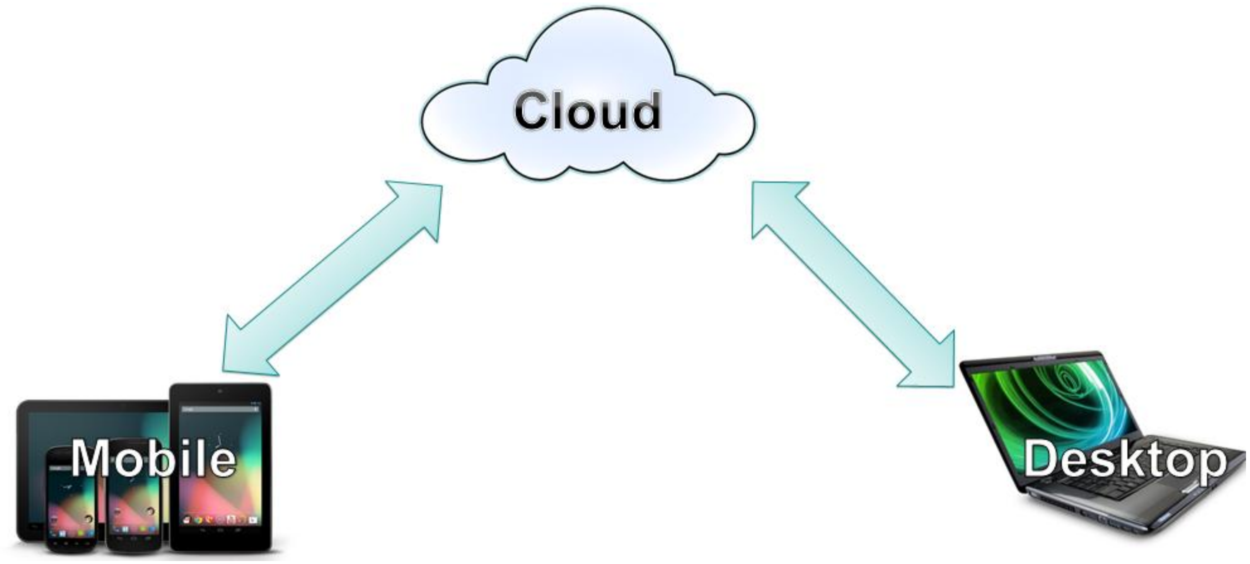


Although this sample presents a transit sign, the idea can be applied to many others scenarios. What would you use this? Control your company assets on the field? Catalog constructions along a roadway? Use geo-referenced images on Civil 3D have many interesting applications.

Proposed architecture and technology

At the beginning of this study, some architectures were considered, but the main idea was collect data with mobile devices (GPS-enabled) and bring it to Civil 3D. The first idea was collect data, store on the mobile then transfer to Civil 3D. This approach is interesting, but restricts the amount of users (devices) collecting data and the speed: just one user (device) at a time, and can only transfer when back to the office (on a desktop with Civil 3D).

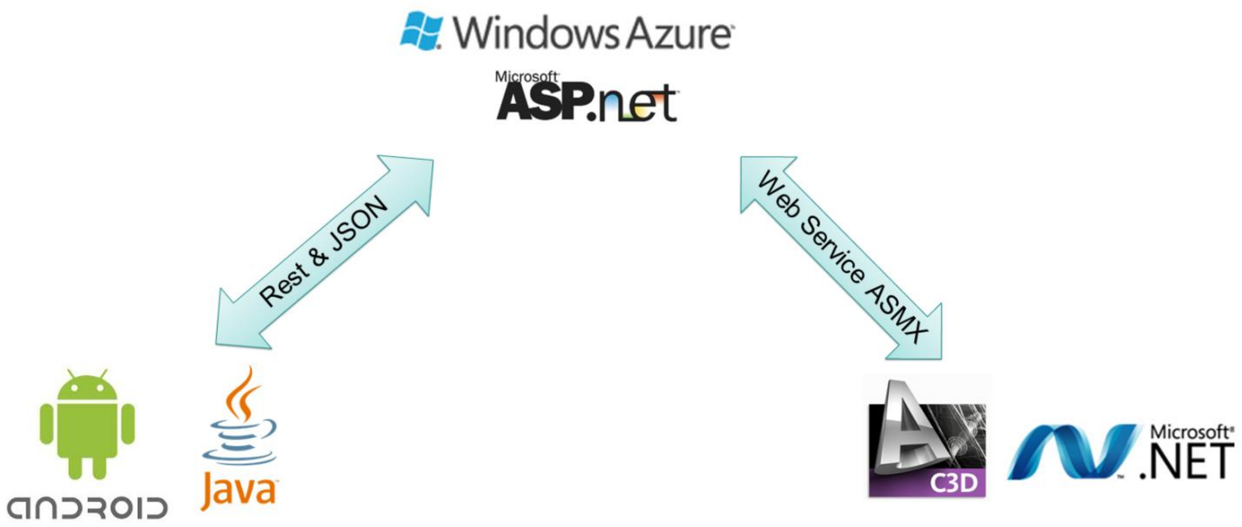
So a more flexible approach came: collect data with mobile, upload to the cloud and, in real time, download to Civil 3D project. The other side is that internet connection is required on the mobile, but in many countries, this is almost not a problem as most users already have internet connection on their phones. Below is an image that summarizes the architecture that will be used. Due time restrictions, a direct connection between mobile and desktop (e.g. via USB) will not be discussed.



There any many available technologies to implement this architecture. Some alternatives will be presented, but it's not limited to the presented names. Also, due a company or market preferences, a more complete/deep review is suggested.

- The Cloud technology is intimately connected to the chosen cloud host. The most famous and reliable are Windows Azure, Google Cloud Platform and Amazon Web Services.
- The Mobile world is dramatically changing, so platforms like Balckberry from RIM and Symbian from Nokia are rapidly decreasing their marked share, so the most relevant alternatives for Mobile are the Android OS from Google, iOS from Apple and Windows Phone from Microsoft.
- The Desktop requires software that can host geo-referenced data. So we might consider Civil 3D and Map 3D.

Now the technology to implement this architecture: due a synergy with Visual Studio and .NET technology, used on Autodesk APIs, the Cloud will use **ASP.NET** hosted on **Windows Azure** service. As an Apple computer is required to develop for iOS (iPhone/iPad), and their license clearly states that their OS cannot be installed on Virtual Machines, and as this implementation will also make use of Civil 3D (Windows), the selected mobile platform was **Android** with the **Java** development technology from Oracle. Due our implementation scenario, this Desktop part will be implemented on **Civil 3D** with **.NET**. The image below presents a graphic representation of these technologies.



The image above also has the connections between the modules. The .NET technology has a good and easy to use mechanism to connect systems across the web: web service. The ASMX technology was used to connect the desktop (Civil 3D) and the cloud (Azure). There is also another technology, SVC, based on Windows Communication Foundation. But after some testing and reading, it's clear that ASMX is too heavy for mobile communication, where the amount of data transferred is connected to the battery use. So a lighter approach with Rest and JSON was selected.

Cloud

Let's start with the Cloud module. Why? Because both Mobile and Desktop modules will connect to it. The Windows Azure cloud host provides services for websites, database with SQL Server, pure store (disk space) and complete virtual machines (that can be accessed with Remote Desktop Access). There are several plans and price ranges, but start as low as $10 US dollars per month (September 2012). This implementation will use a website server, to host the web services, and a data base server to store the data collected.

Website with web services

Visual Studio or Visual Web Express is required to develop a website with ASP.NET. It can be written with any .NET language, but here the C# will be used. There is a SDK for Azure development that will help during deployment: simply download the profile from the website, then inside Visual Studio choose Publish and the profile downloaded.

Web Service for Desktop access

As the desktop side doesn't have severe bandwidth restrictions, the implementation of this side has no special requirements. Just create a simple .asmx service and expose the method with **[WebMethod]** attribute. Note that this method can return arrays and custom types. This implementation will return an array of custom **PointData** objects.

```
public struct PointData
{
  public double latitude;
  public double longitude;
  public string description;
  public string link;
}

[WebMethod]
public PointData[] GetPointData(string projectName)
{
     // implementation here, see source code.
}
```

This method gets the list of points collected for the **projectName**. Note that it returns the location latitude and longitude, the description text and a link to the image hosted on the cloud. This means that Civil 3D drawing will not store the image.

Web Service for Mobile access

This is the tricky part. Mobile access have more restricted bandwidth, even if the carrier provider is very fast, the amount of data transferred is proportional to the battery used. So transfer big chunk of data will cause the battery to end faster.

The Rest protocol consists (in a very simple explanation) in make HTTP (mainly POST and GET) requests to server and get its response as text. A .NET webserver can be adjusted to operate on this approach, just add a modifier to return text on JSON (or any other format). The requests via POST are native.

```
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public string[] GetProjects()
{
     // implementation here, see source code.
}
```

The return must be as string or arrays of string based objects. Numeric values are also supported, but some localized conversions can cause problems, so pay attention. The method above returns the list of projects so the mobile user can select where the image will be grouped.

Below is another method implemented for the mobile client. It is used to post (store) data the collected point on the cloud. As custom data cannot be submitted here (e.g. .NET Image object), only strings and doubles were used. The image must be converted to a string on 64 base, which is a common used solution and methods are available on .NET and Java (that will be used on the Mobile module with Android).

```
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)]
public void PostData(string project,
  double latitude, double longitude,
  string description,
  string imageBase64String)
{
}
```

Database server with SQL Server

It is possible store several data types on SQL Server, and this sample requires text for the description and image for the photo, which are available by default. The latitude and longitude can be stored with numeric values (double), but in order to allow further improvements, it's possible to use geography type only available after install Feature Pack.

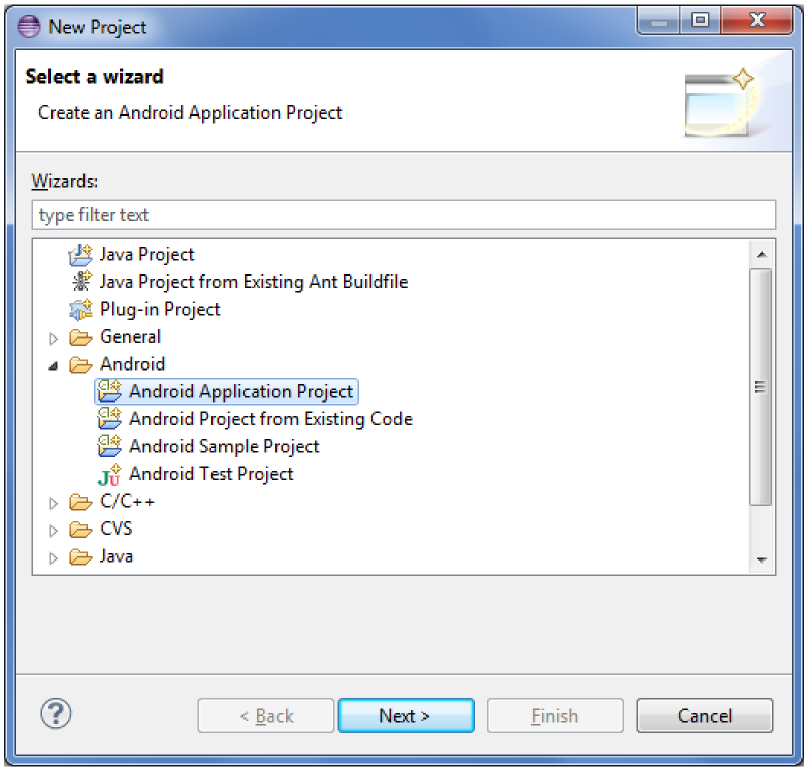
The Azure service can create the SQL Server and connect to the website by a connection string. It also has a good manager interface through web browser or any client connected. This sample will use two tables, Projects and CollectedPoints.

Mobile

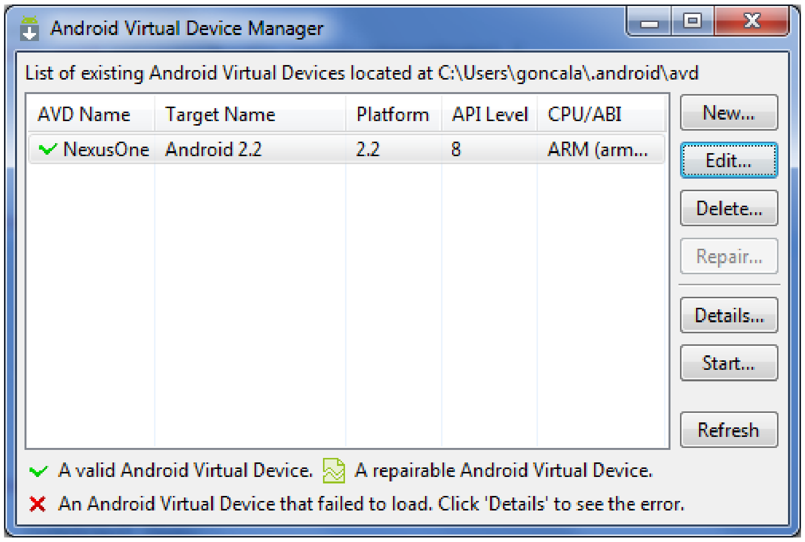
This module was can be developed for any device as the first module is platform independent. As mentioned before, this material will use Android platform from Google, which can be deployed on phones and tablets.

Different from the Cloud module, this requires a Java IDE. Google suggest and offer an interesting Android SDK for Eclipse, which includes the templates required to create projects and a good emulator to test the projects. It also has a package of drivers to debug on an actual device, which is recommended. The image below shows the Android template at the list of templates after we install the SDK.

The emulator, shown below, can be configured for any version of the OS. From their website, the version 2.2 is used on more than 95% of the devices. It is very important to pay attention to this, as there are many changes between 2 and 3, and 3 and 4. This project was first developed for 2 and then migrated to 4 (available on most new phones for sale today), so as a reference point, the code was mainly the same, except for a one security change.



“New Project” wizard with Android template



List of existing Android Virtual Devices located at C:\Users\goncala\.android\avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ NexusOne	Android 2.2	2.2	8	ARM (arm...

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Emulators available at Eclipse after installing the Android SDK

The Android development is based on Activities. Basically any new feature added to the project will be implemented on a separated class that implements this interface. One important change between version 2 and 4 is that Internet requests cannot be made from the main Activity, but to maintain the same code stream for both versions, this sample code added a security feature to allow this connection.

The GPS is tracked at the **LocationChanged** event. There is a strategy to improve the accuracy at Android Develop webpage. To use the location, several permission are required, so please make sure all are specified and the manifest XML. The sample code of this project has all permissions required, below is the relevant part.

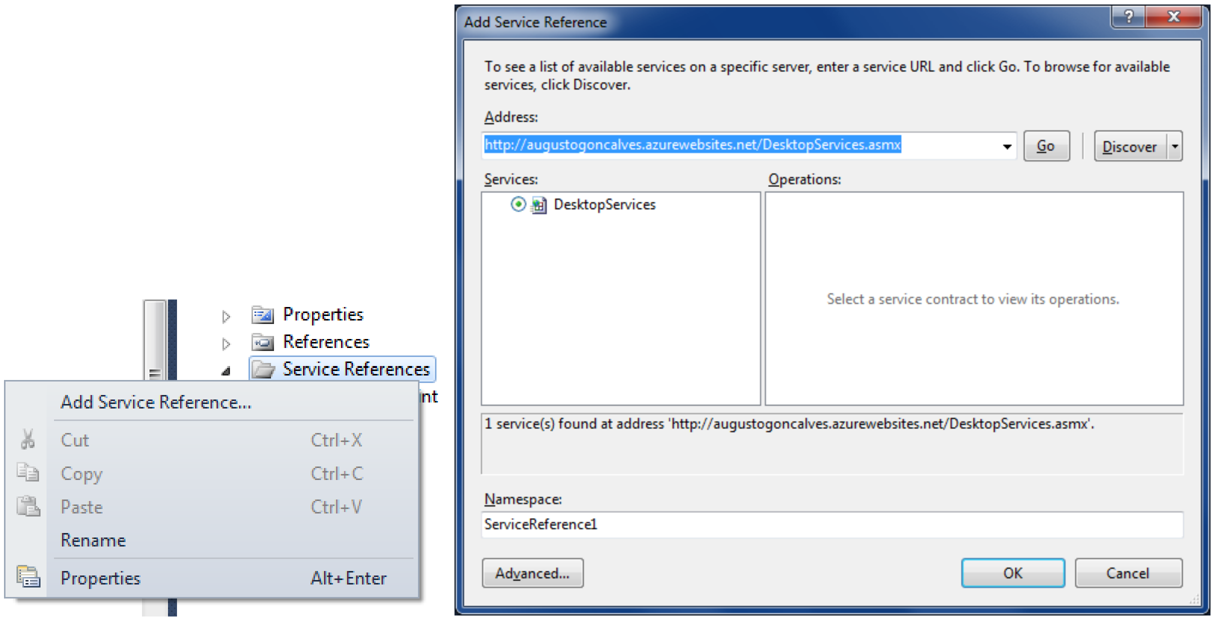
```
<uses-permission android:name="android.permission.ACCESS_GPS" />
<uses-permission android:name="android.permission.ACCESS_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />

<uses-feature android:name="android.hardware.camera" />
```

Now the camera also requires some permissions and features. The sample code above shows it. Like other features, the Camera is an Activity, and we can use the native (built-in camera) or build a custom one. This sample will use the native for simplicity. To use it, just start a camera activity and wait for the response at **onActivityResult** event.

Desktop

This module was implemented using regular Civil 3D APIs. This class will not focus on getting start with this API, but there are some training materials available at the Developer Center, including previous webcasts and blog.

Two features are not usual: connect to the web and convert coordinate units. The first was implemented using the web service feature of Visual Studio, so just add reference to the web service pointing to the cloud location of the .asmx file as shown below. To convert between coordinate systems, more specifically from LL84 (default for GPS coordinates) to the current Civil 3D project, two new references are required: **OSGeo.MapGuide.Geometry.dll** and **OSGeo.MapGuide.Foundation.dll**, both available on any Civil 3D installation folder. Below is a sample code demonstrating the approach to make this conversion. Before execute it, make sure the coordinate system is set on the drawing settings of Civil 3D.



To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://augustogoncalves.azurewebsites.net/DesktopServices.asmx

Services: DesktopServices

Operations: Select a service contract to view its operations.

1 service(s) found at address 'http://augustogoncalves.azurewebsites.net/DesktopServices.asmx'.

Namespace: ServiceReference1

Adding reference to a web reference

```
public static Point3d ConvertPointCoordinate(Point3d point_LL84)
{
  MgCoordinate coord = null;
  if (CSTransform != null && !double.IsNaN(point_LL84.Z))
    coord = CSTransform.Transform(point_LL84.X, point_LL84.Y, point_LL84.Z);
  else
    coord = CSTransform.Transform(point_LL84.X, point_LL84.Y);
  return new Point3d(coord.X, coord.Y, coord.Z);
}

private static MgCoordinateSystemTransform _cstransform = null;

private static MgCoordinateSystemTransform CSTransform
{
  get
  {
    if (_cstransform != null) return _cstransform;
    // get Civil3D current coordinate system
    CivilDocument civilDoc = CivilApplication.ActiveDocument;
    string code = civilDoc.Settings.DrawingSettings. _
                  UnitZoneSettings.CoordinateSystemCode;
    // get the transformation
    MgCoordinateSystemFactory coordSysFactory = new MgCoordinateSystemFactory();
    MgCoordinateSystem coordSys = coordSysFactory.CreateFromCode(code);
    MgCoordinateSystem wgs84Sys = coordSysFactory.Create("LL84");
    // convert the coordinates
    _cstransform = coordSysFactory.GetTransform(wgs84Sys, coordSys);
    return _cstransform;
  }
}
```

Further reading

Some useful Civil 3D® online material:

- Civilized Development blog - http://civilizeddevelopment.typepad.com
  Isaac Rodrigues's Civil 3D API focused blog includes several example codes for .NET

- Civil 3D® Online help with documentation of the entire API

- Civil 3D® Developers Center – http://www.autodesk.com/developcivil
  Training material, recorded presentations and tools.

- Discussion Groups monitored by Autodesk team.

- Information about the Autodesk Developer Network - http://www.autodesk.com/joinadn
  ADN members can ask unlimited API questions through our DevHelp Online interface

- API Training - http://www.autodesk.com/apitraining
  Information about upcoming training classes and webcasts, also download of webcasts

- Watch out for our regular ADN DevLab events. DevLab is a programmers' workshop (free to ADN and non-ADN members) where you can come and discuss your AutoCAD programming problems with the ADN DevTech team.

Conclusion

Thank you for attending this session on Civil 3D, Cloud and Mobile. I hope you found the class enjoyable and valuable. In this handout I have presented information that will be very helpful when redoing the sample.

You have seen how the architecture was idealized and separated into 3 modules, then a summary of relevant information to implement this on your project. Please also review the slide deck (Power Point file) and the code project sample, for web site with ASP.NET, for Android with Java and for Civil 3D with .NET.

I'm also glad you are considering integrate your Civil 3D development with Cloud and Mobile, we believe the best approach is use what's best for each scenario: with our extremely connect technology, the cloud is everywhere and accessible almost any time with mobile devices. Take advantage of that! Good luck!