Data management: Best practices

About this document

The information in this document was gathered through our work with a variety of clients and environments in the field. We thank our customers and partners for sharing their experiences and insights.

What's covered	Best practices, tools, and procedures for deploying and managing geospatial data on Google mapping products and other geospatial platforms.
Primary audience	Technical project managers, solution architects, developers, data managers, and deployment engineers with experience in GIS.
Key takeaways	• A data management plan is crucial for any deployment. • It's important to understand the characteristics of the different types of geospatial data—vector, imagery, terrain, and raster—and how that data is stored and processed. • Google recommends a migration methodology that includes 1) preparing your data for use with Google mapping products another GIS, 2) testing preprocessed data, 3) publishing data, and 4) updating data
Release notes	11 April 2013 - Initial release of document
Feedback	Google values your feedback. Send your comments about this guide to enterprise-assets-feedback@google.com.

Introduction to data management

A data management plan is crucial for any deployment. Storing, updating, and accessing your data quickly and easily are the core goals of a data management strategy.

Spending time up front to define your data management strategy ensures that your project will be successful and creates a robust foundation to support future efforts.

This document provides data management strategies based on:

1. Types of data.
2. Data migration methodology developed by Google.

Types of data

Spatial data comes in a variety of formats, including vector, imagery, terrain, and raster. Depending on the type and complexity of the data you're working with, there are multiple options and considerations to take into account to ensure an optimal deployment. In this section, we'll examine the different data types and provide general recommendations and best practices for each format.

Working with vectors

Vector data consists of points, lines, and polygon features that are defined by a series of X and Y coordinates (and Z for 3D), along with associated attributes. Vector data is typically used to represent features such as roads, parcels, lakes, rivers, streams, building footprints, and points of interest.

One of the primary benefits of vector data is that it's easy to customize and change the symbology in order to support your mapping applications. In addition, vector features typically have a rich set of associated attributes that can be queried against to provide information to the end user. In this section, we cover some best practices when working with vector data.

Server-side vs. client-side rendering

When working with vector data, it's important to define how you plan to serve the information to your end users. Although the source data can be stored in a variety of formats, such KML, GeoJSON, Shapefile, and WKT, how that data will be served to your applications can be very different. The two high-level techniques for presenting this data include server-side and client-side rendering.

Server-side rendering involves converting your vector data to raster graphics, such as JPG, PNG, and GIF images, and returning those graphics to your application. With this approach, you can render hundreds of features on the server and return a single image to the client tier. The primary benefit is that you can drastically reduce network traffic between your application and the server, as you don't need to download the raw vector data. In addition, if your vector data doesn't change often, you can preprocess that data and cache it on the server tier for significant performance gains.

Advantages	Disadvantages
• Reduced network traffic • Render thousands of point, line, and polygon features without impacting network performance • Ability to cache tiles for direct access	• Querying data requires additional server requests • Need to rebuild tiles when data changes • Difficult to change symbology on the fly

Formats:

- Google Earth Enterprise Maps & Globe Services
- Google Maps Engine Services
- WMS
- WMTS
- JPEG, PNG, and GIF (file types)

Client-side rendering involves streaming vector data directly to your application and rendering at that level. Once data is on the client tier, symbology can be changed on the fly without additional server requests. Also, if attribute information is included with the vector responses, no additional server requests are needed to query that data. Depending on the size and complexity of the vector data, however, you need to take into account network bottlenecks and client-side CPU and memory limitations to render the features. Technologies such as GZIP compression and Canvas make client-side rendering faster than in the past.

Advantages	Disadvantages
• Symbology on the fly • Direct access to attribute information • Interactive feature editing	• Network performance • As feature count and complexity increases, performance reduces • Memory considerations / destroy features no longer used

Formats:

- KML / KMZ
- GeoJSON
- GeoRSS
- WKT
- WFS
- SVG, VML, and Canvas (rendering)

Deployment considerations

Here are some key questions to consider when deciding between server-side and client side rendering for your deployment.

- What kind of vector data will you be displaying (points, lines, polygons)?
- How complex is the vector data?

- What is the number of features and number of vertices?
- How often is the vector data being updated? Hourly, daily, weekly, monthly?
- Can any of the line and polygon features be generalized at higher zoom levels?
- Can the data be represented via clustering or using the Heatmap Layer?
- What Google Geo products will be used to process and render the vector data (cloud vs. on-premises)?
- What are the symbolization and display rule settings?

General recommendations

- When possible, use generalization, scale dependencies, and clustering to limit the number of vector features displayed at a given time.
- Only query and display data when necessary. To improve network performance, use AJAX to query data within the map viewport.
- When displaying multiple markers, consider using the MarkerClusterer to consolidate them. Additional details are on Google Maps API site.
- For data that doesn't change frequently, consider preprocessing that data on the server tier for improved application performance.

Use cases

- Store locator: Create an application to display a company's store locations.
    - Use spatial- and attribute-based searching to limit the number of store locations displayed on the map at any given time, as client-side rendering limitations can impact performance as marker count increases.
    - Depending on the number of stores displayed, a client-side or server-side approach could work well here. For example, if a large number of stores will be displayed, consider using server-side rendering; for a smaller number of stores, client-side rendering may be a better solution. Often times, reducing the number of stores displayed on the map at a given time will result in a better end-user experience.

- Asset tracking: Create an application to track a company's delivery trucks.
    - Vehicle locations can be represented by a marker without refreshing the backend map data.
    - Since the data changes frequently, client-side rendering improves application performance.

- Real estate and tax parcel mapping: Create an application to show details of real property.
    - Utilize layers to show parcels, floodplains, utility lines, schools or other features.
    - Since the data doesn't change frequently, server-side rendering improves application performance.
    - Query parcel polygons on the client tier to provide an interactive end user experience.

Working with imagery

Imagery is a raster format that consists of satellite, aerial, and street view data sets. Unlike vector data that is stored as a series of X and Y coordinates, raster data is stored as a matrix of pixel values. Some examples of raster data formats include JPEG 2000, GeoTIFF, MrSID, and ECW. Imagery is not typically updated as frequently as vector data sets, so often times preprocessing and caching imagery data is the preferred approach for serving up to your applications.

Deployment considerations

- How much storage and disk space is required for the project?
    - Disk space and network bandwidth are often the two primary bottlenecks when dealing with large imagery data sets. It's very important during the planning phases of your project to ensure you have ample storage for both the raw imagery and post-processed data. The cost of storage is relatively cheap compared to other infrastructure costs, so try to procure enough space for both existing and future imagery data sets you may acquire.

- How long it will take to process your imagery? How many cores are needed?
    - The length of time to process your imagery data sets is ultimately determined by the amount of CPU cores available and network bottlenecks to access the data.
    - If running an on-premises solution such as Google Earth Enterprise, the faster the CPU cores available to the Google Fusion Server, the faster your data can be processed. Select speed over quantity for CPU cores of the Google Fusion Server, as it is inherently built for single-threaded processing. In addition, select fast storage for the volume that will contain the Asset Tree, as this will have heavy write operations. Commands like *gemaptilegen* and *gevectorfuse* can consume a lot of RAM (easily 8 GB per process in some instances). Select a machine with a lot of RAM if you will be processing large complex vector data sets or creating 2D map tiles.
    - A cloud-based solution like Google Maps Engine doesn't have the same CPU limitations because it uses thousands of Google cores to process data. In this case, network bandwidth (uploading the data to the cloud) is typically the primary factor in determining the time to process your data.

- What format is your source data?
    - Google Maps Engine and Google Earth Enterprise support many of the common imagery formats. When you're planning a deployment, it's important that your data meet the format requirements of the tool you're using. You may need to convert your data to be compatible, for example, from ECW to JPEG2000.

- Is all the data in a common projection and resolution?
    - When working with multiple imagery data sets on a project, having that data in the same projection, resolution, and number of color bands is very important during processing. Google uses GDAL to automatically mosaic a set of images. If an image within that set has a differing number of bands and projection, it will fail during the gdal_mosaic process. Commands like *gdalinfo* can be very useful to diagnose common imagery errors.

General recommendations

- Invest in enough storage to support your existing imagery data sets with room for growth. The ability to visualize historical imagery over time is powerful functionality, but requires significant storage capacity as new data is acquired.
- Define your area of interest and limit the amount of high resolution imagery for other areas of the globe in order to reduce disk space usage.
- Whenever possible, consolidate image files into a single image mosaic to reduce resources and help with imagery data management.
- Use scripts whenever possible to automate image processing on a scheduled interval.

Use cases

- Emergency management: Create an application for an emergency situation.
    - Use current and historical imagery for crisis response and natural disaster recovery.
    - Leverage high resolution imagery to create a single point of reference to help identify key areas of interest for emergency management.
- Agricultural research: Create an application for agricultural purposes.
    - Remote sensing to identify deforestation, vegetation rate, erosion, and land use, just to name a few.
- Recreation viewers: Create an application for users to find recreational opportunities.
    - The ability to see high resolution imagery of hiking trails, ski trails, campgrounds, rivers and streams can be very useful for people planning their next outdoor adventure.

Working with terrain and raster

Terrain data provides 3D representations of a surface based on elevation data. Common formats used for terrain include Digital Elevation Models (DEM), Digital Terrain Elevation Data (DTED), and Triangular Irregular Networks (TIN). LIDAR is a method used to collect terrain data as a point cloud, which can be converted into one of these formats. Many of the same challenges you face when working with imagery data sets holds true for terrain as well.

Deployment considerations

In addition to storage and processing considerations mentioned previously in the Working with Imagery section, these are some additional considerations to take into account when working with terrain and raster data.

- What's the accuracy and resolution of your terrain data sets?
    - When working with multiple terrain data sets, it's important to know the resolution of each file. Common resolutions include 30 meters (1 arc-second), 10 meters (1/3 arc-second), and 3 meters (1/9 arc-second).
    - If your area of interest contains terrain data with varying levels of accuracy, your 3D globe could encounter "cliffs" where the borders of the files meet. Without having the same

accuracy for all files, there's little that can be done to avoid this. From a visualization perspective, reducing the exaggeration of the terrain data is the best way to reduce the appearance of "cliffs" around the borders.
    - The resolution of your terrain data sets also has a large impact on the size. 10-meter DEMs have 9 times more points than 30-meter DEMS, increasing the file size by 9 times.

- What format is your terrain data?
    - Google Earth Enterprise supports a variety of terrain format types. Make sure your data meets one of the format types defined here.

- Is your raster data properly georeferenced?
    - Raster data must be properly georeferenced. For example, the USGS scans topo maps and provides them as Digital Raster Graphics (DRG). Without these images being georeferenced, we would not know where to display them on top of a map or globe. When developing your own custom raster data sets, be sure they are properly georeferenced to points on the earth to avoid issues.

General recommendations

- Whenever possible, try to use terrain that has the same resolution and accuracy for a particular area of interest to avoid "cliffs" at borders.
- Similar to imagery, define your area of interest and limit the amount of high accuracy terrain data for other areas of the globe in order to reduce disk space usage.

Use cases

- Emergency management: Create an application for an emergency situation.
    - Terrain data can be very important to identify key areas of interest for emergency management.
    - For security applications, terrain data can be used for line-of-sight analysis.

- Recreation viewers: Create an application for users to find recreational opportunities.
    - The ability to see elevation data of hiking trails and ski trails can be very useful for users to plan their next outdoor adventure.
    - Virtually explore 3D fly throughs of your planned route.

Data migration methodology

The Geo Enterprise Deployment Team data migration methodology includes four steps:

- Preprocessing: Preparing your data for use with your selected GIS.
- Initial load: Testing preprocessed data prior to uploading it.
- Post-load process: Publishing your data.
- Update & automation: Automating the updates of your data.

Data preprocessing

You'll need to process your data before it's ready to use. Processing changes the original data, so it's essential that you maintain a backup copy of your original data.
After you process your data, you'll need to verify it for accuracy and to ensure it meets the customer requirements.

A best practice for data preprocessing is to document the processes you follow for each type of data and data set, so you can repeat the processes in the future.

Format conversion

Your data must be in a format supported by your GIS platform or application. When converting your files, it's important to do so in a manner that doesn't compromise fidelity or accuracy beyond an acceptable level. Always verify the quality of your converted data.

The process for format conversion is illustrated in this diagram:

Convert a file (or group file, such as a shapefile), from Format A to Format B, using a geo transform tool like GDAL.

- Vector conversion checklist:
    - Verify that the file is complete and can be read in a desktop GIS application, before and after the conversion.
    - Verify that you have all the files (extensions) necessary (like in shapefile).
    - Verify that the file is georeferenced and you know the Coordinate Reference System (CRS).
    - Determine the precision you want on your data, defining the number types (integer, double) and double checking in the output format.
    - Verify that your platform or application supports non UTF-8 character codes.

- Image conversion checklist:

    - Verify that the file is georeferenced and you know the CRS.
    - Determine the number of image bands (standard is RGB, but can be extended) and if you want to maintain the number in the output file.
    - Double check image data bits (8, 12, 24, 32...) and tool compatibility.
    - Double check image color bits and tool compatibility.

- Raster conversion checklist:
    - Verify that the file is georeferenced and you know the CRS.
    - Check the accuracy of the raster data (integer, double) and tool compatibility.

Georeference

Your data must be georeferenced so it can be positioned correctly on the map. If not, the data is liable to be inaccurate, incorrect, or even dangerous. To georeference your data, you'll need to assign it a Coordinate Reference System (CRS) and define where your data is located.

Definitions:

- SRS Spatial Reference System (SRS) or Coordinate Reference System (CRS): A coordinate-based system used to locate geographical entities.
- Map Projection: A method to represent the surface of a sphere or other 3D body on a plane. Map projection is used to represent the Earth on a map.
- EPSG: European Petroleum Survey Group.
- EPSG OGP Geomatics Committee: The committee that defines the standards for CRS.

Best practices:

- Use the correct CRS for your data source.
- Use EPSG-compliant tools to define your CRS.

Geo transform

Once you have a georeferenced asset, it may be necessary to transform it from one CRS to another.

Best practices:

- Document any change from one CRS to another, and the method used.
- Use EPSG-compliant tools to transform from one CRS to another.
- If your GIS application or platform supports different CRS as input, maintain your original CRS. If not, convert all the data to the same CRS if possible.

Data model definition

To work effectively with your data in a GIS application, create a data model that identifies the entities you want to represent and the relationships between them.

There are multiple approaches for defining a data model, for example entity-relationship. It might also be

helpful to create conceptual and physical models in your database. It's strongly recommend that you take the time to create a complete data model early in your project; it will save you time later!

Definitions:

- Data model: A digital representation of a real-world entity, assuming some simplifications and limitations.

Best practices:

- Invest time early in the process to create a thorough data model.
- Validate your data model with your customer to ensure it is complete.

Tools for data preprocessing

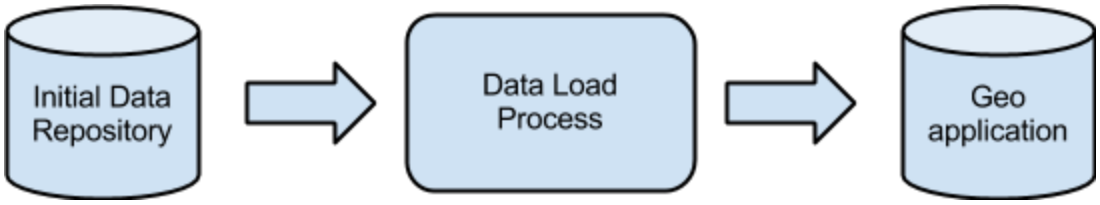
- GDAL
- qGIS
- Grass

Initial data load

After you preprocess your data, you're ready to perform an initial data load.

Define data load procedure

Before loading your data, define a step-by-step data load procedure. It can be a document or an automated script, for example in Python.



Best practices:

- Define step-by-step procedures for uploading the data.
- Remember to consider all types of data (vector, imagery, raster, etc.) and all the exceptions your have (format, CRS, etc.).
- Automate the process to save time and effort.
- For large amounts of data (100s GB to TB), use automated upload tools.
- Check your HD space in your initial and final repositories.
- Check your connection to your geo application repository to estimate the data load duration.

Quality assurance

Perform a thorough assessment of your data to ensure it uploads successfully.

Best practices:

- Check a representative percentage of files for integrity and compare original and final files.
- Verify the data's CRS accuracy using a desktop GIS.
- If your application supports downloading a data set, download the data to verify it.
- Verify fields and data for vector data; pixel value for raster; visual appearance for imagery, etc.
- Create a tool (for example, a script) to check data integrity before uploading it.

Data post-load process

After your data is loaded into your GIS application or platform, there are still a few steps left to complete the process. For example, you may need to generate a tile cache or create map layers.

Post-load processes

Layer creation

Creating a layer typically involves selecting the data you want to display, defining the symbology, determining the scale at which it will be visible (for example 1:e or map level), defining a name for the layer, then publishing.

Map Creation

A map is a collection of layers, with a defined behavior (scale, symbology, etc.) grouped to show a representation of a real-world location. You can define a basemap (e.g. Google Maps) and one or more business layers to help you on this effort.

Tiles pregeneration

Some geo applications or platforms allow you to pre-generate tiles of your layers or maps. Generating a tile cache enhances performance because the server or platform doesn't have to generate these images for each request--the images are already generated.

Quality assurance

When your data is ready, it's time to run some functional and load tests to ensure it meets your customer's needs.

Data update (automation)

When data changes, you need to upload it to the application again. This can happen frequently (for example, a car changing position and sending GPS coordinates to the application) or infrequently (for example, if you acquire new imagery from a country). The most efficient way to handle these updates is through an automated process.

Data update procedure

Follow these best practices when defining your automated data update procedure.

Best practices:

- Define the data to be uploaded and the frequency of updates.
- Identify a person who can provide the data and ensure its accuracy.
- Estimate the size of data to be uploaded and ensure that your systems have ample capacity.
- Identify a person to provide quality assurance on the data.
- Specify the days and times your data will be uploaded and ensure that it fits with your production cycle.