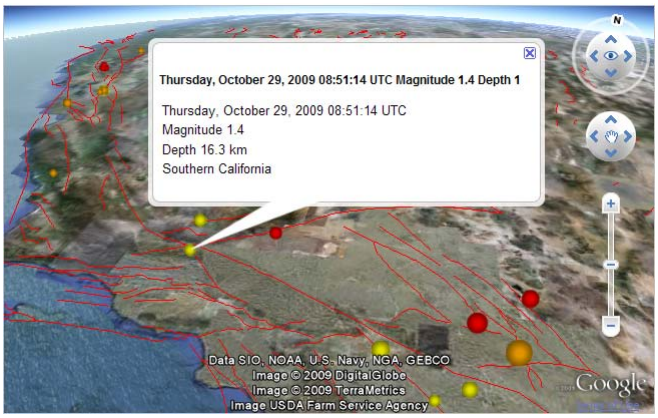
Sample Script

Building Dynamic Web Geomashups

MicroImages has prepared a demonstration of an automated, regularly-scheduled geomashup application that downloads updated geospatial data from the Internet, processes and combines the data with other geospatial data, and posts the result on a web page for viewing in the Google Earth browser plug-in (see the Technical Guide entitled *Geomedia Publishing: Today's Earthquakes in California and Nevada*). The key component of this application is a custom processing script written in the TNT Geospatial Scripting Language (SML) that is run hourly by the TNTmips Pro Job Processing System.

The data in this example are global earthquake epicenter locations and associated attributes. These data are posted and continuously updated as a comma-separated text file by the U.S. Geological Survey Earthquake Hazards Program (at http://earthquakes.usgs.gov/eqcenter/catalogs/). This sample SML script (canvquakes.sml, excerpted on the reverse of this page) takes advantage of the many capabilities in SML:

- the HTTP_CLIENT class for connecting to web servers and downloading data;
- the MIE classes for importing data; and
- the KML class, which allows a script to render a layout containing various geospatial layers into a KML file that anyone can then view in the Google Earth desktop application or in a browser with the Google Earth plug-in.



Google Earth plug-in displaying KML earthquake overlay produced by the sample script and the native Google global imagery. The feature balloon information shown when an earthquake epicenter marker is clicked is tranferred to the KML file from the TNT DataTip settings.

Processing Steps in SML Script to Create a Google Earth Geomashup with Earthquake Data (canvquakes.sml)

1. Connect to USGS web site and download current 24-hour global earthquake epicenter file (comma-separated text).
2. Import point data in text file to a temporary TNT vector object and store earthquake magnitude, depth, and other parameters in a database table with 1 record per point.
3. Extract epicenter points and records for the specified range of latitude and longitude to a new temporary vector object.
4. Add string expression field to the point database table and assign expression to create a multiline string listing earthquake attributes for each point for use as a DataTip.
5. Add extracted epicenter vector object to an existing page layout containing a vector layer with major active fault lines.
6. Modify settings for earthquake epicenter vector layer to specify the database field for point DataTips.
7. Set earthquake epicenter point styles by script; symbol size is based on magnitude, color based on depth.
8. Render the modified layout to a KML file referenced by the sample web page.
9. Create text file with date and time the KML file was created (text to be shown on web page).

The box above right summarizes the script processing steps, from downloading the text file to rendering a layout with styled earthquake points and reference fault lines to a finished KML file. The earthquake epicenter points in the TNT layout are converted to placemarks in the KML file. The script takes advantage of several automated features of the KML rendering procedure in SML. First, any vector element attribute information that is set for viewing in a DataTip in the TNT products is automatically converted to a "description" for the resulting placemark in the KML file. Google Earth shows this description in a pop-up "feature balloon" when each placemark is clicked. The sample script therefore sets up a DataTip for each earthquake epicenter point using a string expression field it creates in the vector point database table. The expression for this field creates a listing of multiple earthquake attributes that are read from various other fields in the table.

The KML rendering procedure in the SML script also automatically assigns 3D marker symbols (sphere or cube) from the Google Earth marker library to any TNT vector points that are styled using the TNT predefined point symbols (circle or box). The sample script therefore assigns epicenter point symbols using a query that references preset styles in a style object stored in a reference TNT Project File. Marker size is set to vary with earthquake magnitude, and the marker color is determined by the depth of the earthquake below the ground surface.

Automated running of this sample script on a repeating schedule in the TNTmips Pro Job Processing System is discussed in the TechGuide entitled *System: Scheduling Automatically Repeating Jobs*. (over)

Many sample scripts have been prepared to illustrate how you might use the features of the TNT products' scripting language for scripts and queries. These scripts can be downloaded from www.microimages.com/downloads/scripts.htm.

Script Excerpts for canvquakes.sml

1 Download USGS daily global earthquake summary file

```
class STRING address$ = "earthquakes.usgs.gov";
class STRING url$ =
   "http://earthquakes.usgs.gov/eqcenter/catalogs/eqs1day-M1.txt";
class STRING textfile$ = _context.ScriptDir + "/Download/eqs1day-M1.txt";
class HTTP_CLIENT http;
clear();

http.SetTimeOut(15);
err = http.Connect(address$, 80);          // connect to the web server

http.DownloadFile(url$, textfile$);        // download the text file
```

2 Import the comma-delimited text file of earthquake epicenters to points in a vector object

```
class RVC_OBJECT tempfile;
tempfile.MakeTempFile(1);                  // set up new vector object in tempfile

class RVC_OBJITEM fqObjItem;               // object item for vector with global
class RVC_DESCRIPTOR fqDescript;           // daily earthquake epicenters
fqDescript.SetName("GlobalQuakes");
fqDescript.SetDescription("");

fqObjItem.CreateNew(tempfile.GetObjItem(), "VECTOR", fqDescript);
   // set up class for import/export of vector from/to text; the
   // class method to import the object takes an RVC_OBJITEM
   // rather than an RVC_VECTOR class instance.
class MieTEXTVECTOR mieTV;
   // set existing format file that records all import settings including
   // additional database fields and the coordinate reference system
   // for the earthquake data
class STRING formatFilename$ =
   _context.ScriptDir + "/resources/eqFormat.fmt";
mieTV.FormatFilename = formatFilename$;

err = mieTV.ImportObject(textfile$, fqObjItem);
printf("mie ImportObject returned %d\n\n", err);
```

3 Copy epicenter points within the specified latitude/longitude range to a new temporary vector object

```
class RVC_VECTOR CaNv_Quakes;              // create new temporary
                                           // vector object for the
CreateTempVector(CaNv_Quakes, "Planar");   // California-Nevada
                                           // epicenter points
                                           // query string to select epicenter points
                                           // within desired latitude-longitude range
class STRING ptQry;
ptQry =  "if (CLASS.Lat > 32.5 && CLASS.Lat < 42.0 && CLASS.Lon > -124.75
&& CLASS.Lon < -114.0) return true;";

class RVC_VECTOR GlobalQuakes;             // class instance for the global earth-
                                           // quake vector object, needed for the
                                           // VectorCopy Elements function
   // open the global earthquake vector object
GlobalQuakes.OpenByName(fqObjItem.GetFilePath(),
                        fqObjItem.GetObjectPath(), "Read");
   // use query to copy epicenters from global earthquake vector
   // to temporary California-Nevada epicenter vector object
err = VectorCopyElements(GlobalQuakes, CaNv_Quakes,
                         "RemExRecords", ptQry);
printf("VectorCopyElements returned %d\n\n", err);

GlobalQuakes.Close();                      // close the global earthquake vector object
```

5 Read page layout with major fault lines and add the earthquake epicenter vector to it.

```
class STRING layoutfile$ = _context.ScriptDir +
                  "/resources/QuakeResources.rvc";   // filename for the layout
class GRE_LAYOUT layout;
layout.Read(layoutfile$, "FaultLayout");

class GRE_GROUP group;
group = layout.FirstGroup;                 // get handle for the single group in the layout
   // add the earthquake epicenter vector to the group in the layout
class GRE_LAYER_VECTOR quakeLayer;
quakeLayer = GroupQuickAddVectorVar(group, CaNv_Quakes);
```

7 Set the earthquake epicenter point styles by script using a previously saved style script file that uses point styles already created and saved in a style object in a reference Project File. These styles use the predefined filled circle symbol with different colors assigned by earthquake depth; Render to KML automatically converts these predefined circle symbols into a shaded sphere symbol in Google Earth.

```
   // set previously-created style object with point styles to use
class STRING styleFile$ = _context.ScriptDir +
      "/resources/QuakeResources.rvc";
class FILEPATH styleFilepath(styleFile$);
class RVC_STYLE ptStyles;
ptStyles.OpenByName(styleFilepath, "PtStyles.STYLE", "Read");
quakeLayer.StyleObject = ptStyles;
   // set up earthquake point styling by script;  read style
   // script from saved file to avoid syntax check prob-
   // lems with Table.Field references in the query
class STRING ptStyleQry;
ptStyleQry = TextFileReadFull( sprintf("%s/resources/QuakeStyle.qry",
                                       _context.ScriptDir) );
quakeLayer.Point.StyleMode = "ByScript";
quakeLayer.Point.Script = ptStyleQry;
```

8 Render the layout to a KML file.

```
print("Rendering updated layout to KML.");
class DATETIME dt;                         // get current datetime
dt.SetCurrent();                           // and convert to UTC
dt.ConvertToUTC();

class STRING datetime$;                    // string with datetime for web page
datetime$ = dt.GetString();
datetime$ += " UTC";
printf("update datetime: \n", datetime$);
   // open text file and write datetime string (overwrite)
class STRING dtFilename$ =  _context.ScriptDir + "/datetime.txt";
class FILE dtFile = fopen(dtFilename$, "w");
fwritestring(dtFile, datetime$);
fclose(dtFile);

class STRING kmlName$ = _context.ScriptDir + "/eqs1day_canv.kml";
class FILEPATH kmlPath(kmlName$);          // KML file information

class KML kml;                             // render the modified layout to KML
kml.SetPath(kmlPath);
kml.SetLayout(layout);
kml.SetResolution(450);                    // target resolution 450 m
kml.Write();

CaNv_Quakes.Close();                       // close temporary objects
tempfile.Close();
```