

Sample SML Tool Script Run Browser

The Run Browser script provides an example of a custom script used to launch an external application. The web browser was chosen as the example program because it is the one type of external program that all clients are most likely to have so you can run the script. The function that launches the external application is the same for any file type; it simply uses the file name provided to determine

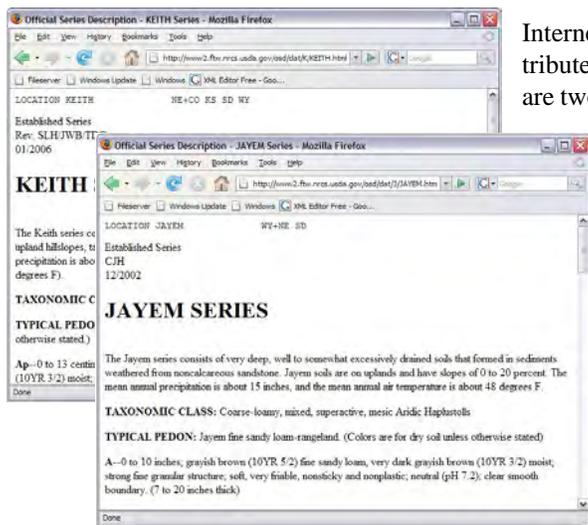
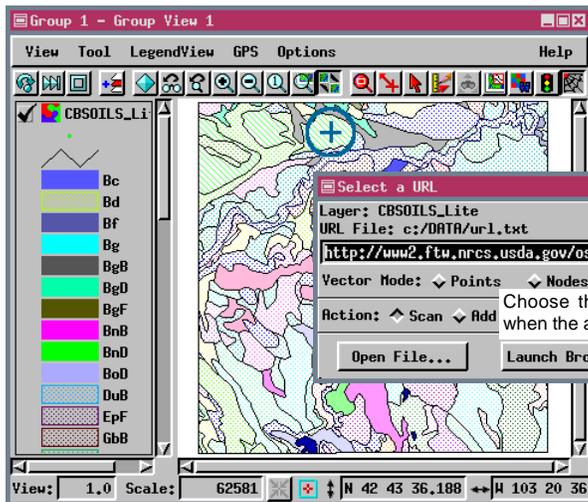
which application to launch. For example, file names that end in .ppt will launch Microsoft PowerPoint.

The Run Browser script lets you select elements in a vector or cell values in a raster layer, choose an associated URL for a web site to visit, then launch your Internet browser and go to the selected site. The associations between element attributes or cell values and URLs are specified in a separate text file, which means there are two files needed to use this tool (urls.sml and url.txt). The text file specifically lists the name and description for each object with URL links. Any number of objects can be listed in the file, but if the active layer in the group or layout does not contain one of the objects named, you get a "File not found" message instead of a list of URLs. To get results when you use this tool without altering the sample text file, you need to have either CBSOILS_LITE from the CB_SOILS Project File (CB_DATA folder) or CLS_MAXLIKE from the BERCRPCL Project File (BEREA folder) as the active layer. Both Project Files are found with the litedata on your TNT products CD-ROM or with the TNTlite datasets on MicroImages' web site.

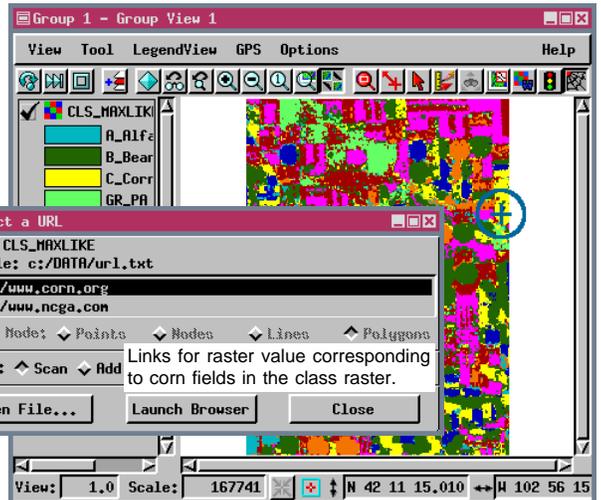
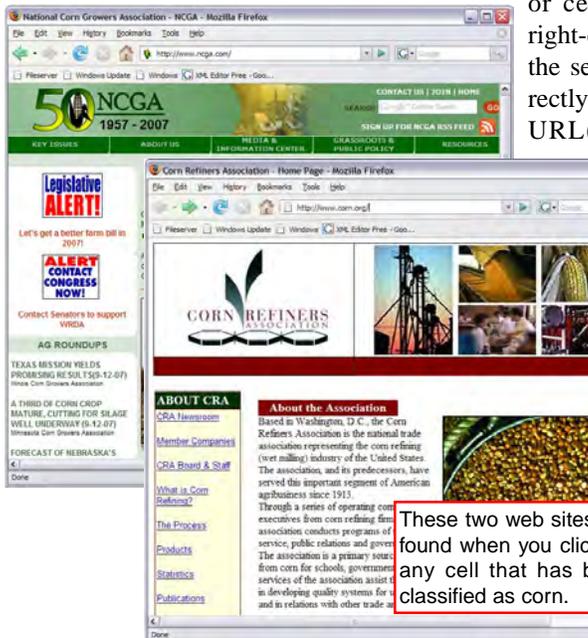
You can associate one or more URLs with each attribute or cell value. You can also associate the same URL with different attributes if desired. The sample text file associates polygons of soil type KeB, JmD, or Bd in CBSOILS_LITE with pages about the corresponding soil type at National Research Conservation Service's web site. You could, of course, associate all the soil types with appropriate pages. To use the tool, left-click on the polygon or cell desired then right-click to confirm the select tool is correctly positioned. The URL(s) associated

with where you clicked appear in the Select a URL window. Choose the desired URL, then click on the Launch Browser button. Your Internet browser will open (if not open already) and go to the designated web site and specified page.

You can easily add URL links for your own objects to this file. Toward the bottom of the Select a URL window there is an Action panel that lets you choose between Scan and Add. Switch to Add, and for a vector, left-click and right-click on an element of the type you want to link to. You are then prompted to select a table and field for the attribute and finally, to enter the URL you want elements with the selected attribute value to link to. For a raster, you are simply prompted to enter the URL to add for the cell value you clicked on. The necessary text with the proper syntax is entered in the url.txt file. You can then toggle on Scan for the action, select the cell or polygon, and go to the designated web site.



can also associate the same URL with different attributes if desired. The sample text file associates polygons of soil type KeB, JmD, or Bd in CBSOILS_LITE with pages about the corresponding soil type at National Research Conservation Service's web site. You could, of course, associate all the soil types with appropriate pages. To use the tool, left-click on the polygon or cell desired then right-click to confirm the select tool is correctly positioned. The URL(s) associated



Macro and Tool Scripts can be created using SML in any TNTmips process that uses a View window (Options / Customize from the View window menu bar). These scripts are then available from an icon, which you select or design, on the toolbar. Sample scripts have been prepared to illustrate how you might use these features, which are available only in TNTmips 6.4 or later, to assist with specific tasks you perform on a regular basis. If possible, the full script is printed below for your quick perusal. When a script is too long to fit on one page, key sections are reproduced below. The sample Tool Script illustrated can be downloaded from the SML script exchange at www.microimages.com/downloads/tool¯o.htm.

Reference File for Run Browser Script (url.txt)

```
[CBSOILS_Lite : (TNTlite) Part of Crow Butte soil type polygons]
[poly CLASS Class KeB]
http://www2.ftw.nrcs.usda.gov/osd/dat/K/KEITH.html
[poly CLASS Class JmD]
http://www2.ftw.nrcs.usda.gov/osd/dat/J/JAYEM.html
[poly CLASS Class Bd]
http://www2.ftw.nrcs.usda.gov/osd/dat/B/BANKARD.html
```

object name and description

URL

element type and attribute value specified

```
[CLS_MAXLIKE : Class raster from 6_06, 7_30, & 10_10 (Green, Red, NIR6)]
{1}
{3}
http://www.microimages.com
```

cell values linking to following URL(s)

```
http://www.wheatworld.org
http://www.oznet.ksu.edu/wheatpage/
{5}
http://www.jeffersoninstitute.org/pubs/drybeans.shtml
{6}
http://www.ag.ndsu.nodak.edu/media/beets/sgrbeet.htm
{7}
http://www.corn.org
http://www.ncga.com
{8}
http://www.animalrangeextension.montana.edu/articles/Forage/General/Growing-alfalfa-hay.htm
```

URLs linked to cell value 7

Partial Script for Run Browser (urls.sml)

```
proc cbLayer() {
  if (Group.ActiveLayer.Type == "Raster") {
    vectorLabel.Sensitive = 0;
    pointButton.Sensitive = 0;
    nodeButton.Sensitive = 0;
    lineButton.Sensitive = 0;
    polyButton.Sensitive = 0
  }
  else {
    vectorLabel.Sensitive = 1;
    polyButton.Sensitive = 1;
    lineButton.Sensitive = 1;
    nodeButton.Sensitive = 1;
    pointButton.Sensitive = 1;
  }
  cbRedraw();
}
```

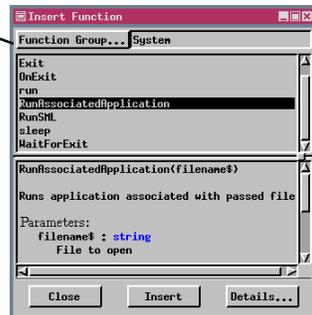
sets window options when active layer changes

```
proc cbOpen() {
  filepath$ = GetInputFileName(filepath$, "Open URL file", "txt");
  cbRedraw();
}
```

Open File button functions

```
proc cbGo() {
  local string url$;
  url$ = list.GetItemAtPos(list.GetFirstSelectedPos());
  if (list.SelectedItemCount > 0 and url$ != "No URLs found!" and url$ != "Type not supported!"
  and url$ != "No element found!" and url$ != "File not found!")
  RunAssociatedApplication(url$);
}
```

Launch Browser button functions



The RunAssociated Application function launches whatever application would be used if you double-clicked on the file on your desktop. If the file name ends in .doc, it will launch Microsoft Word; if the file name ends in .pdf, it will launch Adobe Acrobat or Acrobat Reader, whichever would be used if you double-clicked on the file on your desktop.

unhighlights selected element if active layer is vector

```
proc clearHighlight() {
  if (Group.ActiveLayer.Type == "Vector") {
    local class VECTORLAYER vl;
    vl = Group.ActiveLayer;
    if (mode$ == "point") {
      vl.Point.HighlightSingle(1);
      vl.Point.HighlightSingle(1, 3);
    }
    else if (mode$ == "node") {
      vl.Node.HighlightSingle(1);
      vl.Node.HighlightSingle(1, 3);
    }
    else if (mode$ == "line") {
      vl.Line.HighlightSingle(1);
      vl.Line.HighlightSingle(1, 3);
    }
    else {
      vl.Poly.HighlightSingle(1);
      vl.Poly.HighlightSingle(1, 3);
    }
  }
}
```

```
proc cbClose() {
  pointTool.Managed = 0;
  DialogClose(form);
  if (setDefaultWhenClose) {
    setDefaultWhenClose = false;
    View.SetDefaultTool();
  }
}
```

Close button functions

```
proc cbModeChanged() {
  if (pointButton.Set == 1) {
    mode$ = "point";
  }
  else if (nodeButton.Set == 1) {
    mode$ = "node";
  }
}
```

changes vector selection mode

```
else if (lineButton.Set == 1) {
  mode$ = "line";
}
else mode$ = "poly";
}
proc cbActionChanged() {
  if (scanButton.Set == 1) {
    action$ = "scan";
  }
  else action$ = "add";
}
proc cbToolApply(class RegionTool polyTool) {
  list.DeleteAllItems();
```

sets Action mode

right mouse button click to confirm selection does the following

clears the list

```
local string url$, layerName$, temp$, temp2$, item$, element$, table$, field$, value$;
local class FILE reffile;
local class LAYER layer;
local numeric numTok, i, j, num, start;
local class POINT2D point;
local class StatusHandle status;
local class StatusContext context;
layer = Group.ActiveLayer;
```

sets up local variables

```
if (layer.Type == "Raster" or layer.Type == "Vector") {
  point.x = pointTool.Point.x;
  point.y = pointTool.Point.y;
```

keep going if active layer is raster or vector

get coordinates of selected point

```
# Set up layer, object, layer name, and point transformations.
if (layer.Type == "Raster") {
  point = TransPoint2D(point, View.GetTransLayerToScreen(View, layer, 1));
  DispGetRasterFromLayer(rv, layer);
  layerName$ = "[" + rv.$Info.Name + " : " + rv.$Info.Desc + "];"
}
else if (layer.Type == "Vector") {
  point = TransPoint2D(point, View.GetTransViewToScreen(View, 1));
  point = TransPoint2D(point, View.GetTransMapToView(View, layer.Projection, 1));
  local class VECTORLAYER vl;
  vl = layer;
  DispGetVectorFromLayer(vv, layer);
  layerName$ = "[" + vv.$Info.Name + " : " + vv.$Info.Desc + "];"
}
reffile = fopen(filepath$);
```

```
start = 0;
while (!feof(reffile)) {
  url$ = fgetline$(reffile);
  start += 1;
  if (url$ == layerName$)
    break;
}
if (url$ != layerName$)
  list.AddItem("File not found!");
```

determine that active layer is named in reference file

```
url$ = "";
while (!feof(reffile)) {
  temp$ = fgetline$(reffile);
  url$ = url$ + temp$ + "\n";
  if (temp$ == "")
    break;
}
```

retrieve URL choices for the active layer

```
fclose(reffile);
```

close the reference file

(see urls.sml for full script)