# KML Service

It is possible to publich two kinds of KML services via ArcGIS Server, a dynamic service where the KML output is created dynamically and a service where a pregenerated KML file is publiched.

Since we’re working with a lot of features, it can only be recomented to publich a pregenerated KML file.

## Optimizing ArcMAP for KML

The E-PRTR dataset contains a lot of points, which I have split up into sector defined group layers. Feature layers is used to create KML folders containing a list of all the features. The key KML settings of a feature layer are:

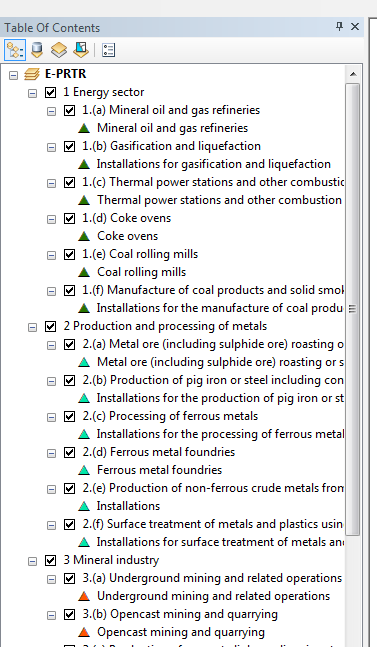
* The Layer Name property, which is used as the folder name.
* The Layer Description property, which is used as the popup content for the containing folder.
* The Symbology, which is used to create a KML symbol.
* The Label expression, which is used to name each feature.
* The Definition Query, which can be used to limit which features are included in the KML representation.
* The HTML Popup properties, which are used to define the popup content for individual features.

These two links to ESRI webhelp describes some of the primary KML settings, others are useful as well.

<http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Preparing_your_map_documents_in_ArcGIS_for_KML_publishing>

<http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Translating_ArcGIS_information_to_KML>

## Creating the E-PRTR mxd document for the KML network link

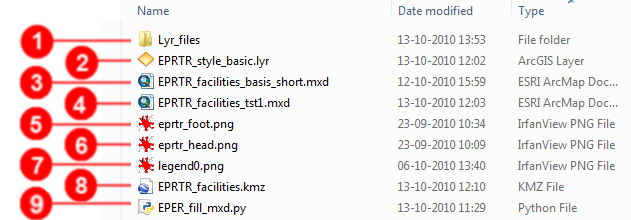
The EPRTR mxd document used for creating the kml file is rather complicated. It contains nine group layers representing the facility sectors containing 45 layers all in all. Each layer has its own name, unique query definition and separate style with title, but shares the same data source, field's definitions, html popup definition etc.

To minimize the maintenance complexity a solution based a python script is made.

The procedure in short:

1. The basic lyr file is edited so the data source etc. is correct
2. The python script is edited so the paths and filenames are correct
3. The base mxd file with predefined group layers is opened
4. The python script is loaded into the python dialog
5. The script is started by hitting the enter key and is completed when the Map2KML procedure successfully completes
6. The kmz file is opened with WinZip – In WinZip the legend0.png file deleted and the tree images; legend0.png, eprtr\_foot.png and eprtr\_head.png is added – done
7. Click on the kmz file to open in GoogleEarth and control that it is correct
8. Upload to the server as an ESRI KML network link

## Files

Pt. there is one folder and eight files in the SVN folder:

1. Folder with symbol lyr files
2. The important basis lyr file where the data source, fields aliases etc. are set
3. A basis mxd file excluding EPRTR sections
4. The mxd file output from the latest successful run
5. The html popup footer image
6. The html popup header image
7. The legend file which replaces the one in the kmz file
8. The kmz file output from the latest successful run
9. The python script which is loaded into ArcMAP and executed

## The basic lyr file

Heres a list of the basic lyr settings:

|  |  |
| --- | --- |
| **General tab:**  Layer name and visibility is later replaced by the script. Descriptions and credits are not. |  |
| **Symbology tab:**  The symbology is replaced by the script. |  |
| **Fields tab:**  It is important that the selection fields are visible and their aliases correspond to the variables used in the HTML popup xslt. |  |
| **Definition Query tab:**  This is where the layer input is filtered. In the current run it is filtered by the ReportingYear equals 2008. |  |
| **Labels tab:**  I chose not to implement labels, as I find them confusing when there are so many features lying so close. But labels are possible to set here. |  |
| **Joins & relates tab:**  I have set up two joins, one for country names and one for activity codes. The fields supplied by the joins are used in the HTML popup. |  |
| **HTML Popup tab:**  This is of course the important tab. We need to set the ‘Show content..’ checkbox and chose ‘As a formatted page based on XSL template’. The XSL input is described here under. |  |

## The XSL template

The XSL basically defines a HTML output (Most of the XSL is to be seen in the text box, the full XSL can be found in the MXD document). The HTML (XSL) is loaded per feature so the bigger and more complex it is, the longer load time and slower performance we end up with.

In the header I have defined most styles

<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org /1999/XSL/Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format">

<xsl:template match="/">

<html><head>

<style type="text/css">

img {height:20px;width:30px;margin-right:2px;}

a{text-decoration:none;color:#8B4513;font-weight:bold;}

..

</style>

</head>

<body leftmargin="0" topmargin="0" bottommargin="0" rightmargin="0">

<font face="Verdana,Arial,Helvetica,sans-serif" color="#222222" size="10px" >

To optimize the kmz file size and to minimize dependency to other resources, the image references are all to embedded images. In the xslt the image is defined by the name only, like eprtr\_head.png. After the kmz file is created, the eprtr\_head.png image is added to the file via WinZip.

I've defined a collection of variables that reference the feature attributes;

<xsl:variable name="frIDCol" select="//Field[FieldName='FacilityReportID']/FieldValue"/>

The variables are used as input and in *if* and *choose* - *where* clauses, so that the fields are only shown if data exists.

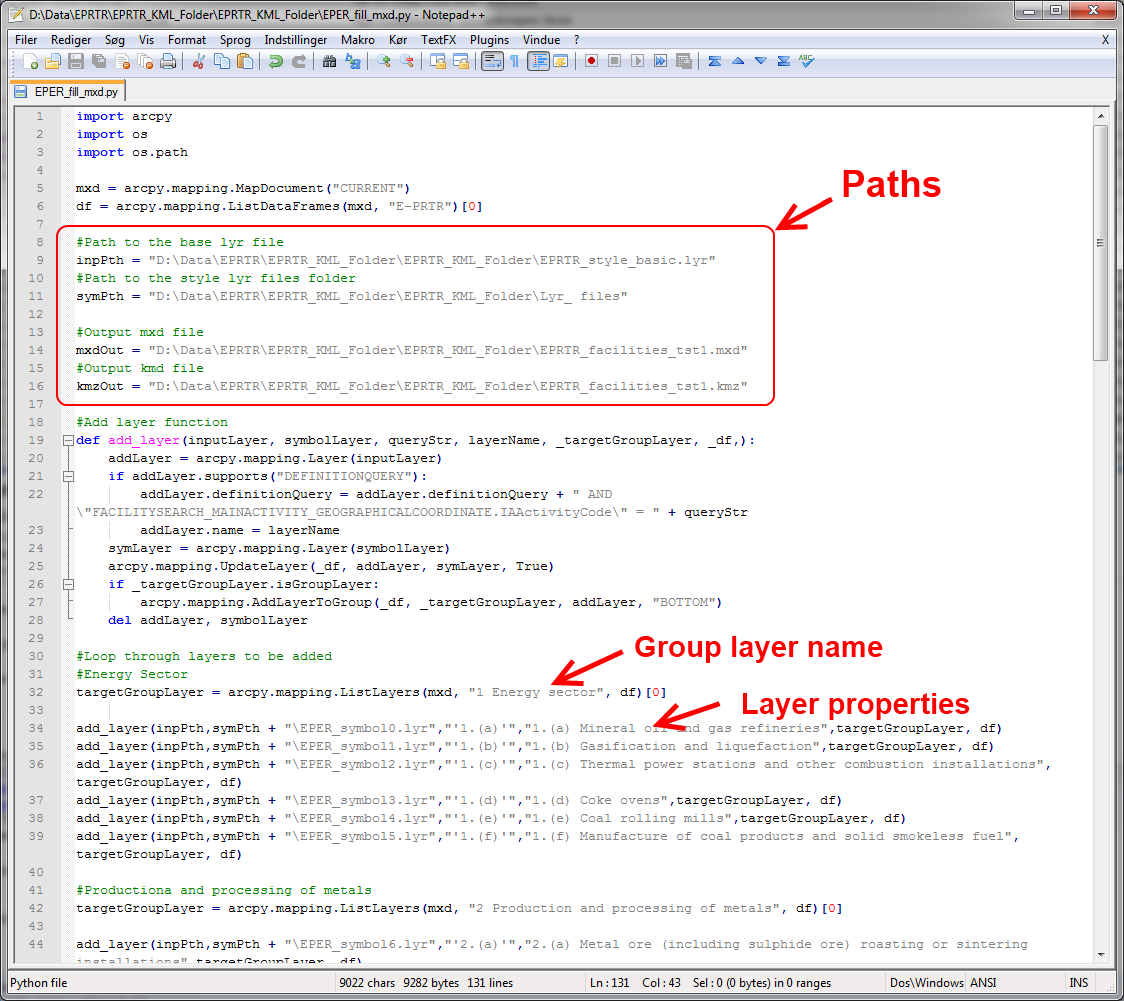
Notice that it is the field alias that is referenced!

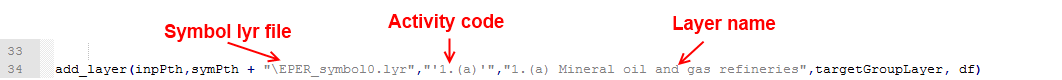
The xslt can be found in the EPRTR\_style\_basic.lyr file.

## The Python script

The script is quite simple. For each of the 45 facility section activities the EPRTR\_basic\_style.lyr is copied and placed in the correct group layer, renamed, filtered and restyled. The restyling is based on the symbol lyr's found in the lyr\_files folder. After the layers has been added, they're set to visible and refreshed. Then a copy of the mxd is saved and a kmz file created.

It is off course important that the EPRTR\_basic\_style.lyr is valid, that the paths are correct and that the query string is correct. The query string will differ whether the data source is ArcSDE or filegeodatabase.



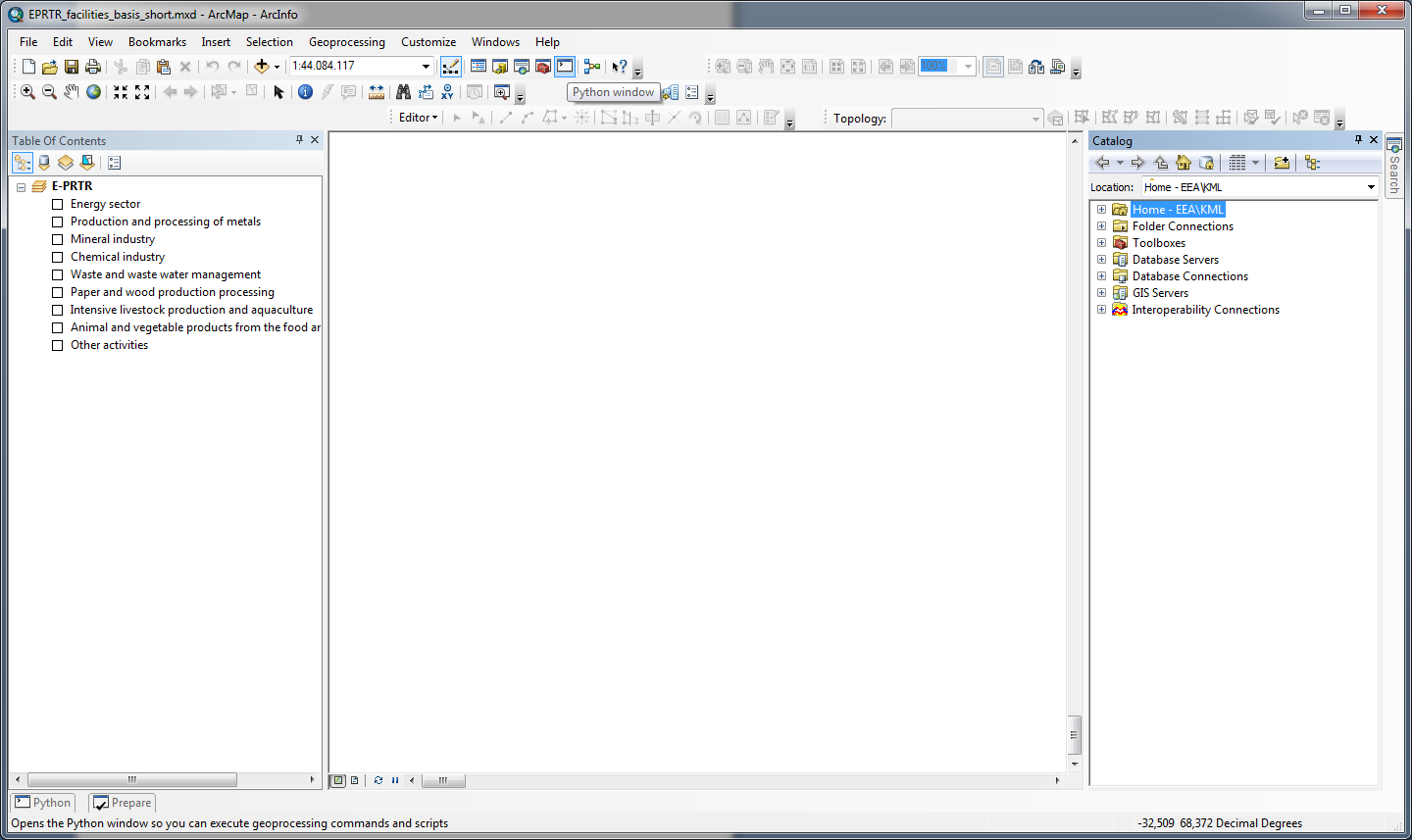


The targetGroupLayer names have to correspond to the group layer names in the EPRTR\_facilities\_basis\_short.mxd.

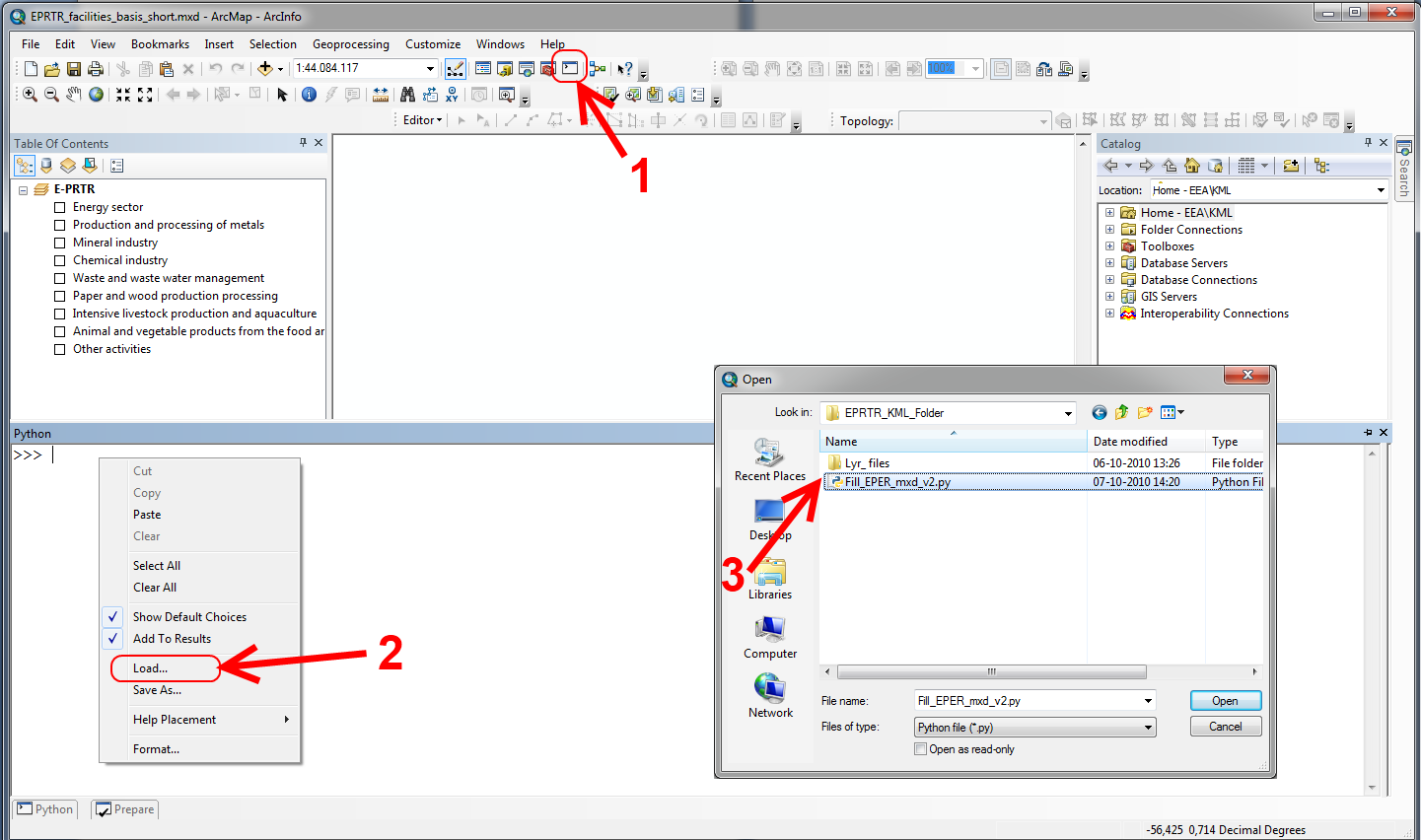
targetGroupLayer = arcpy.mapping.ListLayers(mxd, "Energy sector", df)[0]

## The base mxd document and execution of the python script

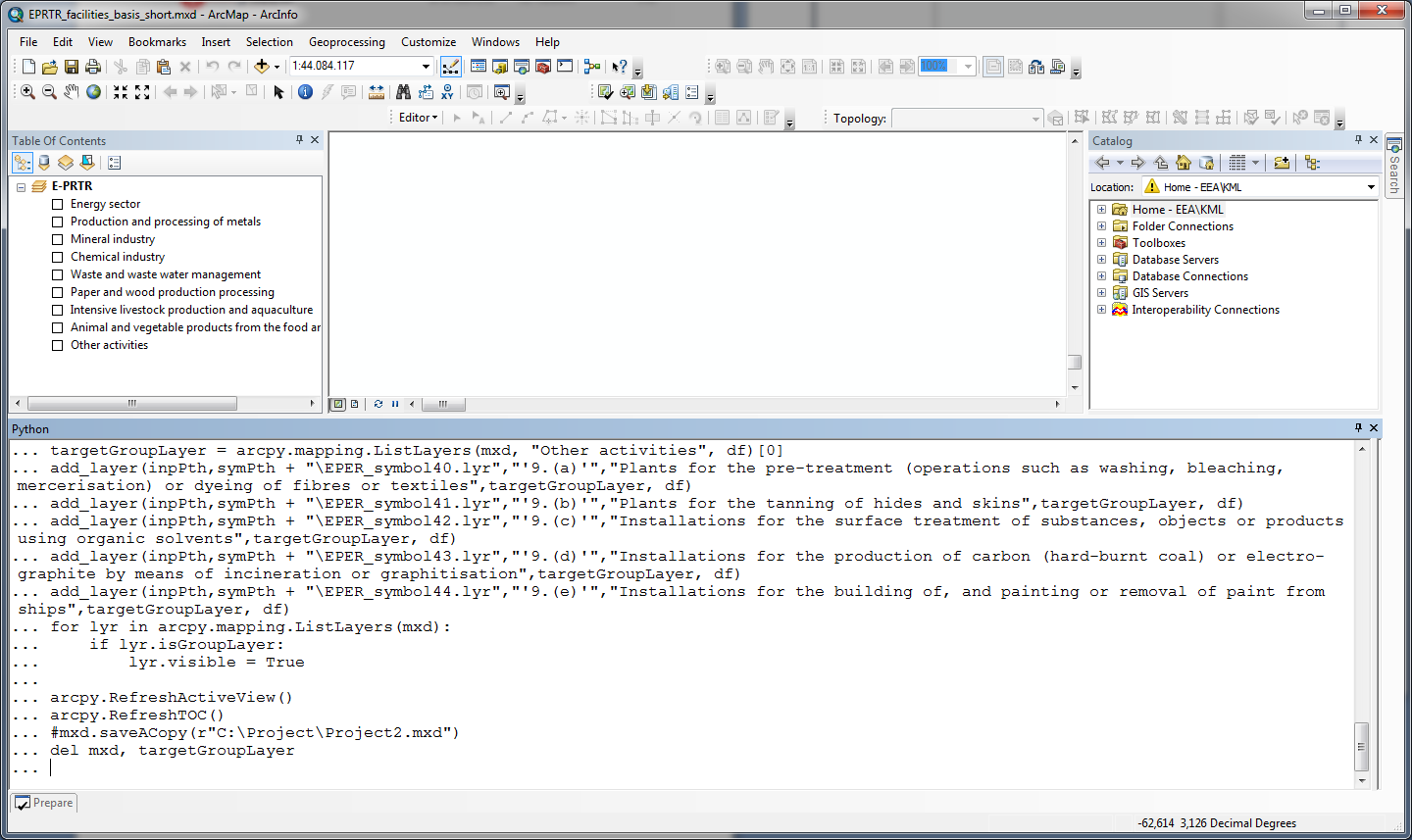
The EPRTR\_facilities\_basis\_short.mxd contains only the empty group layers and an empty legend control (only visible in the layout view).



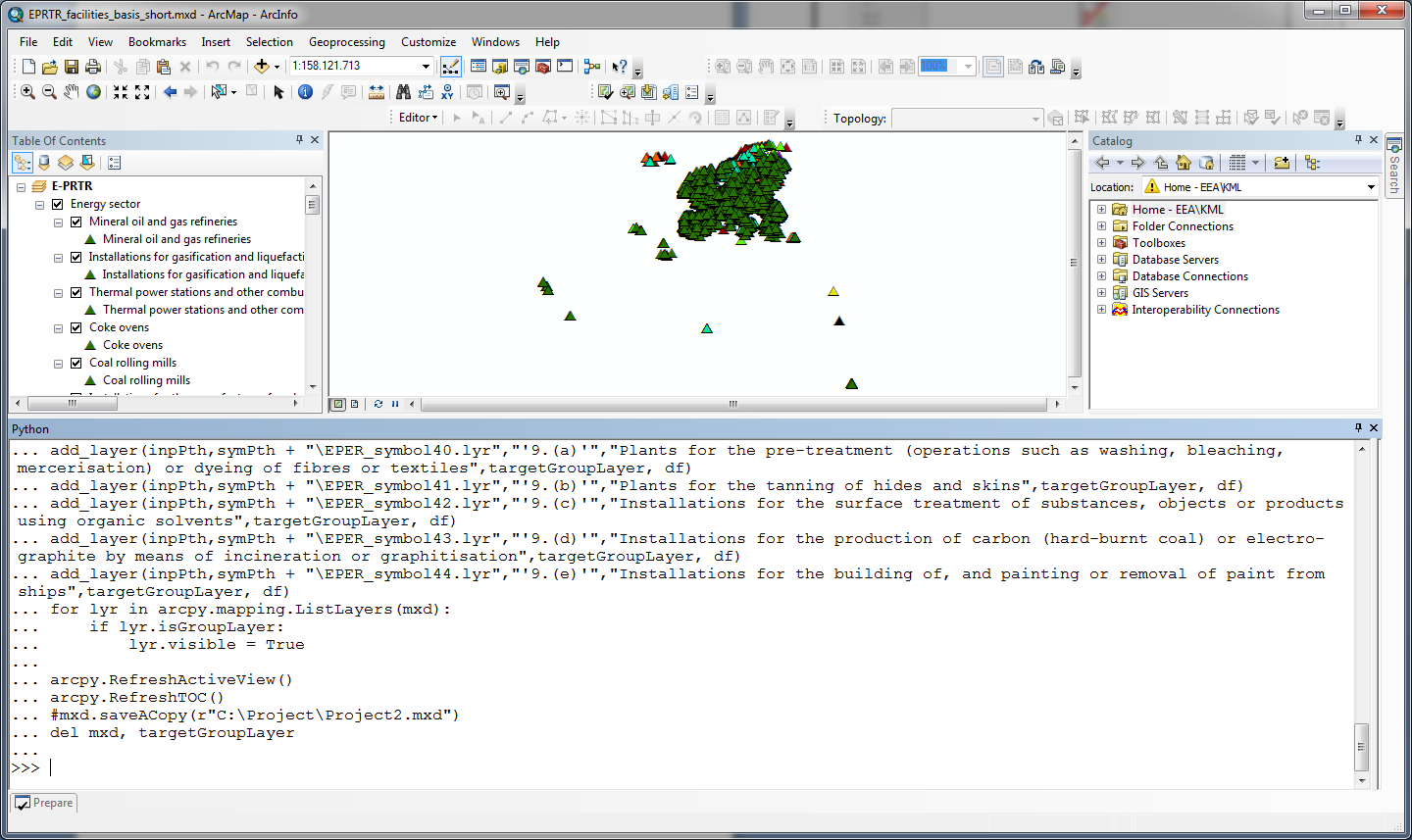
Open the Python dialog (1), right click and choose *Load*(2) and browse to the EPRTR\_fill\_mxd.py file.



When the script is loaded hit the *Enter* key and the script will be activated (be patient, it takes a few seconds to before the curser icon changes).



The script is done when the MapToKML process is done.



NOTICE! Close the mxd without saving – a copy is allready made and if you save, a new EPRTR\_facilities\_basis\_short.mxd has to be made.

## Edit KMZ file

## The last step is to replace and add images to the kmz file. The kmz file is a standard zip document containing the doc.kml file and a lot of images. Open (Not extract) the kmz file with WinZIP or similar program. Remove/delete the legend0.png file and then add the tree images; legend0.png, eprtr\_foot.png and eprtr\_head.png from the solution folder.

Close the WinZIP and click the kmz file to open it in GoogleEarth.

Check that the legend is correct, that the placemark icons on the map is correct and that the the info balloons look correct. Last you can check that the links to the EPRTR facility details view works.

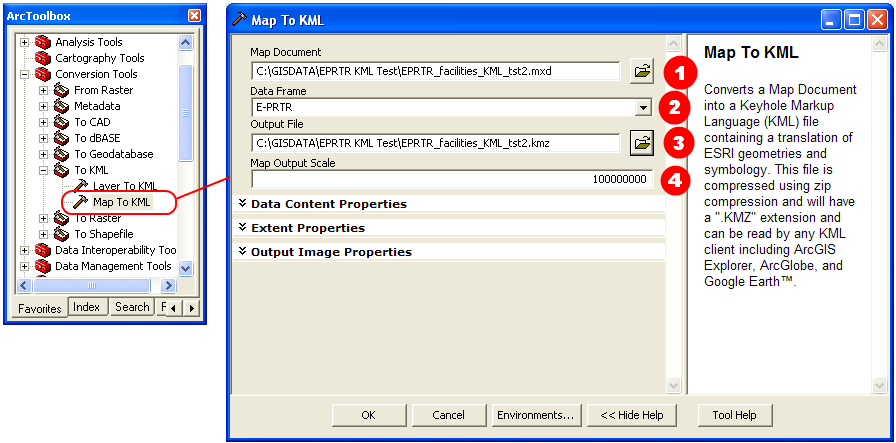
## Create a legend image

I created a new legend image this way:

1. In ArcMAP I added one layer pr group layer to the legend control to get the icons right.
2. Then I used the 'convert to graphics' function to turn the legend in to a graphics layer
3. Then I ungrouped the graphic items and changed the text to the group layer names.
4. Then I exported the map to an image file.
5. In Paint.NET I selected the legend area, copied it into a new image and saved it as legend0.png, which is the name of the legend image in the kmz file.

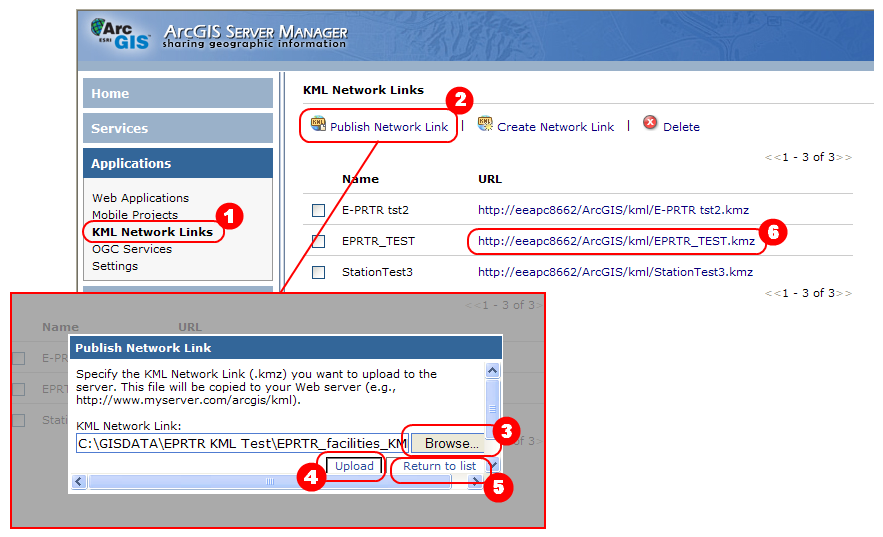
### Map to KML

The tool is now embedded in the python script, but I'll leave the description - just in case.

Browse to the ArcMAP file (Remember to save!! Unsaved changes won’t be recognized) (1). It will probably suggest which dataframe to use (2). Browse to the folder where the kmz file is saved (Notice that the filename will be used as the name for the kml network link)(3). The output scale has to be set, choose 1 for no limits (4). The process can take up to 10 minutes.

## Publish the network link

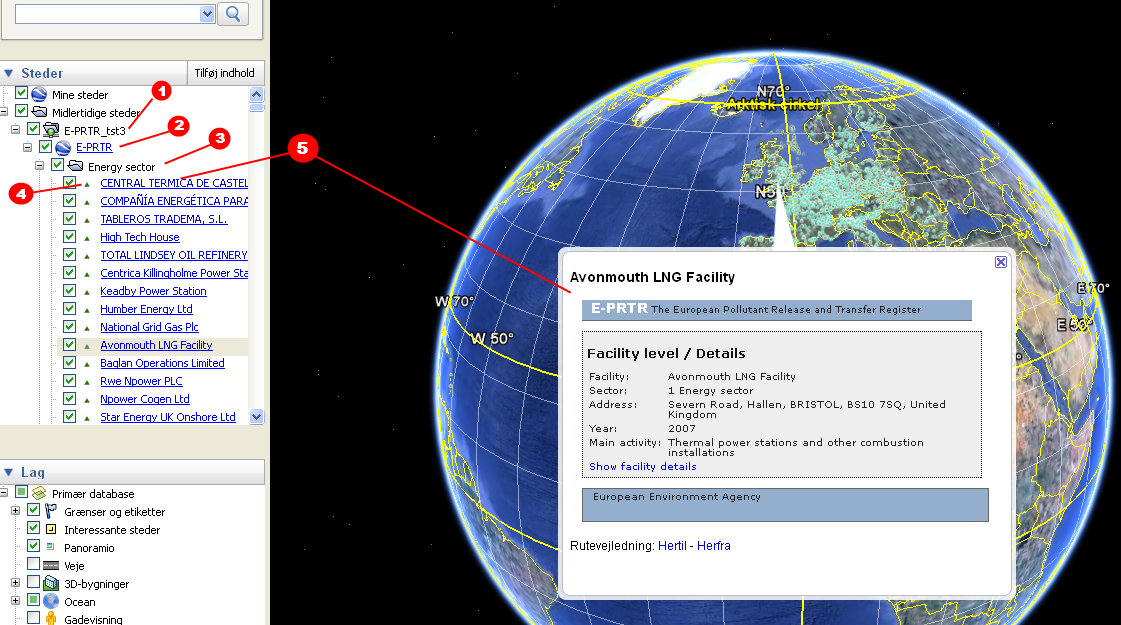
For the kmz to work in Google, you have to create a kml network link.



In ArcGIS Server Manager chose ‘Application’ menu and click on the ‘KML Network Links’ (1). To publish a new link based upon our kmz file, click ‘Publish network Link’ (2). In the popup dialog, browse to the kmz file (3), click ‘Upload’ (4) and then click ‘Return to list’ (5). To test the kml network link, click on the newly created link (5).

**Notice!** The name of the network link is set to the same as the filename. You can’t change, edit or restart the link, only delete and upload / create again. As written earlier, you can publish a network link to a kml service, but its performance is non existing.

## Behaviour in Google Earth



In Google Earth the first container is named after the name of the network link (1) and if you rightclick and chose properties, the description, extent etc. is taken from the kmz file. The child container is named after the ArcMAP Dataframe (2), and the properties here are taken from the dataframe as well. All the sub-containers here (3) corresponds to the layers under the dataframe and is named here after. All the items under the sub-container correspond to features in the layer. After the show / hide checkbox, a layer style icon is displayed (4). If you click on the icon Google Earth will zoom to the feature. The following link (5) is named after the value found in the layer primary display field. Clicking on the link the info balloon is shown and double clicking makes it zoom to the feature.

## History of experience

ERROR: The popup url <http://prtr.ec.europa.eu/Popups/FacilityDetails.aspx> was wrong

SOLUTION: The URL was changes into <http://prtr.ec.europa.eu/PopupFacilityDetails.aspx>

ERROR: Lost the FacilityReportID column with the 'Import' of FACILITYSEARCH\_MAINACTIVITY\_GEOGRAPHICALCOORDINATE to a filegeodatabase.

Reason: FacilityReportID is actually OBJECTID, which must be a design error.

What happened: Got confused and used the FacilityID instead – obviously an error

NOTICE: Used the Main Facility activity names from the LOV\_ANNEXIACTIVITY table, but that was a mistake, because the names didn't correspond to the ones used at the EPRTR website.

NOTICE: Tried to create sub group layers into the sector layers. It took some time to adjust the script, but in the end it worked fine in ArcMAP. Unfortunately we exceeded the levels of child folders in kml. The folders were shown but the features were not, so back to original script.

NOTICE: The auto generated legend by ArcMAP is useless.

I created a new this way:

1. In ArcMAP I added one layer pr group layer to the legend control to get the icons right.
2. Then I used the 'convert to graphics' function to turn the legend in to a graphics layer
3. Then I ungrouped the graphic items and changed the text to the group layer names.
4. Then I exported the map to an image file.
5. In Paint.NET I selected the legend area, copied it into a new image and saved it as legend0.png, which is the name of the legend image in the kmz file.

NOTICE: Uncompressing and compressing a kmz file, you need to make sure that the doc.kml is on top. Instead of extracting and rezipping the file, you just have to open it in WinZIP and remove / add whats needed.