GlobalMapper

SCRIPTING REFERENCE v19

Blue Marble GEOGRAPHICS

Mind the gap between world and map

www.bluemarblegeo.com

Contents

Import/ Open Data

Layer Management

General Overview

Global Mapper script files allow the user to create custom batch processes that make use of the functionality built in to Global Mapper. From a script, one can import data in any of the numerous formats supported by the software, reproject that data if desired or perform analysis on it, and export it to a new file. Scripts can be run from within the application interface using the **File> Run Script...** option. They can also be automatically run in the background by double clicking the *.gms file or called from the command line on a machine with Global Mapper installed.

Syntax

Global Mapper script files consist of a series of command lines. Each command line begins with a **COMMAND**. A series of parameter/value pairs should follow the command. These pairs should be written as **parameter**=**value**. No spaces should exist before or after the equal sign.

Individual parameter/value pairs should be separated by spaces. If a pair requires spaces internal to the value, quotes may be used around the entire value. For example, for a filename with spaces, the pair could look like *FILENAME="c:\\my documents\\test.tif"*.

Commands, parameters and values are case-independent, but by convention, and in this guide, they are written in all uppercase.

Command lines typically consist of one line each. To extend a command to another line, use the backslash character (\) at the end of the line. There are a few exceptions to this, including the DEFINE_PROJ and DEFINE_SHAPE commands and the looping functionality provided by the DIR_LOOP_START and DIR_LOOP_END commands.

Boolean Values

Many parameters in the scripting language are boolean, meaning that they have two states only of either on or off.

The values listed through-out this guide are **YES** to enable the option and **NO** to disable it, but there are a number of accepted values to specify the state:

- YES can be represented by "**YES**", "**Y**", "**TRUE**", "**T**" , "**1**" , or no value (see below).
- NO can be represented by "**NO**", "**N**", "**FALSE**", "**F**", or "**0**".

Parameters that expect a value of YES or NO to enable or disable functionality can (starting with v13.1) be enabled with just the parameter name. So rather than saying FLAG_PARAM_NAME=YES, you can just say FLAG_PARAM_ NAME to get the same behavior as specifying yes.

Wildcards and variables

Some parameters accept values including wildcard characters.

- *** (an asterisk) represents a wildcard of any length.
- *?* (a question mark) represents a single character wildcard.

Parameters can also except variables as values or parts of values. Variables are delimited by *%VARIABLE%*. Some variables are built-in but they can also be defined within the script. Custom variables must be defined before they are used in a command (the script is run from top to bottom sequentially). Typically variables are defined at the top of the script foe ease of use. If a variable is defined with the same name as a previously defined variable, it is replaced with the new value for the rest of the script.

see "DEFINE_VAR" on page 22 for more information.

Comments

Any lines that begin with the forward slash character (/) are considered comments and are ignored by the script processing engine. This means that you can use C or C++ style comments like // or /* at the start of your line.

> **Note:** For user created syntax highlight common in text editors and shared example scripts from users
> see the Global Mapper User Forum

Scripting Command Quick Reference

- ADD_MEASURE_ATTRS - Adds/Updates Measure Attributes to Features in a Layer
- APPLY_FORMULA - Applies a Formula to Loaded Raster Layers to Create a New One
- ASSIGN_TYPE - Deprecated, Use EDIT_VECTOR Instead
- CALC_ATTR - Calculate a New Attribute Value Based on Existing Attribute(s) and a Second Value
- CALC_ATTR_FORMULA - Calculate a New Attribute Value Based on a Formula Combining Existing Attributes
- CALC_VOLUMES - Calculate the Volume of Areas Using Current Elevation Data
- CALC_VOLUME_BETWEEN_SURFACES - Calculates the volume between two elevation grids
- COMBINE_LINES - Combines Connected Line Features Into New Lines or Areas Based on Attribute Values
- COMBINE_TERRAIN - Combines Two Loaded Terrain Layers to Generate a New Terrain Layer
- COPY_ATTRS - Copies Attributes from One Type of Features to Another Spatially
- COPY_LAYER_FILES - Copies the Base Files for Loaded Layers to a New Disk Location
- CROP_AREAS_TO_LINES - Split or crop area features based on line features
- DEFINE_LAYER_STYLE - Define a Layer Style for Later Use
- DEFINE_PROJ - Define a Projection for Later Use
- DEFINE_SDB_CONNECTION - Define an Spatial Database Connection
- DEFINE_SHADER - Define an Elevation/Slope Shader for Later Use
- DEFINE_SHAPE - Define a Shape (i.e. Polygon) for Later Use
- DEFINE_TEXT_FILE - Define Embedded Text File with Features to Load with IMPORT_ASCII
- DEFINE_VAR - Define a Variable for Later Use
- DEFINE_VAR_TABLE - Define a Table of Variable Values for Lookup
- DIR_LOOP_END - Ends a Loop of Commands Over Files in a Directory
- DIR_LOOP_START - Start a Loop of Commands Over Files in a Directory
- EDIT_MAP_CATALOG - Creates or Edits a Map Catalog
- EDIT_VECTOR - Edit Loaded Vector Features that Match a Type/Name/Attribute Query
- EMBED_SCRIPT - Runs Another Script File Within This Script
- EXPORT_ANY - Automatically Use Proper Export Command Based on Target TYPE
- EXPORT_CLOUD - Export data to cloud (Amazon s3)
- EXPORT_ELEVATION - Export Elevation Data to a File
- EXPORT_METADATA - Export Metadata for a Layer to a File
- EXPORT_PACKAGE - Export Data to a Global Mapper Package File
- EXPORT_PDF - Export Data to a PDF File
- EXPORT_PDF3D - Export Data to a 3D PDF File
- EXPORT_RASTER - Export Raster and Elevation Data to a File
- EXPORT_VECTOR - Export Vector Data to a File
- EXPORT_VECTOR_SPATIAL_DB
- EXPORT_WEB - Export Vector Data to a File
- FORCE_EXIT - Forces Global Mapper to Immediately Exit with a Return Code
- GENERATE_CONTOURS - Generate Contours from Elevation Data
- GENERATE_ELEV_GRID - Generates an Elevation Grid from Loaded 3D Vector Data
- GENERATE_EQUAL_VAL_AREAS - Generates Area Features from Equal Values in Elevation/Terrain Layers
- GENERATE_LAYER_BOUNDS - Generates a Layer with Bounding Area Features for each Loaded Layer

- GENERATE_PATH_PROFILE - Generate a 3D Path Profile and Save it to a XYZ File
- GENERATE_POINTS_FROM_ELEV_GRID - Generate points at elevation cell centers
- GENERATE_REPORT - Generates a Report on the Loaded Vector Features
- GENERATE_RIDGE_LINES - Generate Ridge Lines from Elevation Data
- GENERATE_VIEWSHED - Generate Viewshed from Elevation Data
- GENERATE_WATERSHED - Generate Watershed from Elevation Data
- GLOBAL_MAPPER_SCRIPT - Script Header Line
- IF/ELSE_IF/ELSE/END_IF - Conditional Execution Based on Variable Values (If/Then/Else)
- IMPORT - Import Data From a File
- IMPORT_ARCHIVE - Import Data From an Archive File (.zip, .tar.gz, etc.)
- IMPORT_ASCII - Import Generic ASCII Data from a File
- IMPORT_CLOUD - Import Cloud Dataset
- IMPORT_DIR_TREE - Import All Data Files in a Directory Tree
- IMPORT_OSM_TILE - Import Tiled (OSM/TMS/Google Maps/Bing Maps) Online Source
- IMPORT_SPATIAL_DB - Import a Spatial Database
- IMPORT_TERRASERVER - Deprecated, Terraserver-USA/MSRMAPS.COM Servers Down as of May 1, 2012
- IMPORT_WMS - Import WMS Layer
- JOIN_TABLE - Joins Attributes from a File to a loaded Vector Layer
- LAYER_LOOP_END - Ends a Loop of Commands Over Loaded Layers
- LAYER_LOOP_START - Start a Loop of Commands Over Loaded Layers
- LIDAR_CLASSIFY - Automatically Classify Lidar Points
- LIDAR_COMPARE - Compare point cloud to control points.
- LIDAR_EXTRACT - Automatically Extract Building Outlines and Tree Points from Lidar
- LOAD_PROJECTION - Loads a New Global Projection From a PRJ File
- LOAD_STYLE_FILE - Loads a Style/Type File (.gm_style)
- LOAD_TYPE_FILTER - Deprecated, Use SET_OPT Instead (Loads a Lidar Filter or Type Filter from a GMF (Global Mapper Filter) File)
- LOG_MESSAGE - Logs a Status Message
- MAP_LAYOUT - Define the Map Layout (Margins, Scale, etc.)
- PAN_SHARPEN - Pan Sharpens a Color Layer with Pan Image (Creates New Layer)
- PLAY_SOUND - Plays a Beep or a Specified Sound File
- QUERY_LAYER_METADATA - Place Layer Metadta Value in a Variable
- RESTORE_LAST_SAVED_VIEW - Restores Last Saved View
- RUN_COMMAND - Runs a Command Line
- SAVE_CURRENT_VIEW - Saves Current View
- SAVE_PROJECTION - Saves the Current Global Projection to a PRJ File
- SAVE_WORKSPACE - Saves Workspace (GMW) File with Loaded Layers
- SET_BG_COLOR - Sets the Background Color
- SET_LAYER_OPTIONS - Updates Display Options of Loaded Layer
- SET_LOG_FILE - Sets the Name of the Log File
- SET_OPT - Sets General Options (Like Position Display Format, Display Options, etc.)
- SET_VERT_DISP_OPTS - Set Vertical Display Options
- SET_VIEW - Sets the Display View
- SHIFT_LAYER - Shifts the location of a layer by the specified distance.
- SHOW_3D_VIEW - Displays the 3D View Window
- SORT_LAYERS - Sorts the Loaded Layers Based on Some Criteria

- SPLIT_LAYER - Splits a Layer Based on an Attribute
- UNLOAD_ALL - Unloads All Currently Loaded Data
- UNLOAD_LAYER - Unloads a Single Layer
- VAR_LOOP_END - Ends a Loop of Commands Over a Range of Values
- VAR_LOOP_START - Start a Loop of Commands Over a Range of Numeric Values
- VIEW_LAYOUT - Define the Multi-View Layout

Special Parameter Types

- Attribute Names
- Cropping Operations to Polygons/Areas
- Gridding/Tiling Operations into Smaller Chunks
- Projection Specification
- Specify Bounding Box for Operation
- Lidar Advanced Filter Options

Common Scripting Tasks

GLOBAL_MAPPER_SCRIPT

The GLOBAL_MAPPER_SCRIPT must be the first command in the file for scripts prior to v18. . Typically, the entire command line will look like:

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
```

You can use the following parameters with this command:

- **VERSION** - specifies the version of the scripting language used. This parameter is required. You should always use VERSION=**1.00**.
- **ENABLE_PROGRESS** - specifies whether or not any progress dialogs should be displayed while this script is processing. This is enabled by default for scripts run in the context of the main map view or when loading workspaces. Scripts run in their own sandbox or from them command line disable progress by default. Use ENABLE_PROGRESS=**NO** to disable the display of any progress dialogs during the processing of this script.
- **GM_VERSION** - specifies the Global Mapper version that a workspace (.gmw) file was saved from. This is in v16.2.2 and later. The format will be like GM_VERSION="16.2.2".
- **TIMESTAMP** - specifies the time that the workspace was saved. This will be saved in the ISO-8601 time format, like TIMESTAMP="2015-06-03T13:08:39Z"
- **SHOW_WARNINGS** - specifies whether or not warning messages should be displayed when you are done loading the workspace/script into the main map view. Use SHOW_WARNINGS=**NO** to disable the display of warnings. Any true ERROR messages will always display.
- **LOG_TO_COMMAND_PROMPT** - specifies whether or not logged messages should be written to the calling command prompt (if script is passed on the command line). If you add LOG_TO_COMMAND_ PROMPT=**"YES"** and you pass the .gms file on the command line, any logged messages will be written to the normal log context as well as the command prompt. Make sure to call global_mapper.exe with 'start /wait' syntax or from a .bat file to ensure that the command line output goes where desired.

- **REQUIRE_WORKSPACE** - name of workspace file that is required to be loaded for this script to run. If a name is provided for this parameter and that workspace is not currently loaded into Global Mapper, the script will immediately abort. This can be used if you have different scripts that you only want to use if other workspaces are active and want to prevent accidentally selecting the wrong script.

SAVE_WORKSPACE

The SAVE_WORKSPACE command saves any currently loaded layers to a workspace (GMW) file. The following parameters are supported by the command.

- **FILENAME** - full path to GMW file to save

EMBED_SCRIPT

The EMBED_SCRIPT command allows you to call another script from within a script or to load a workspace file. This can be useful in many situations. For example, if you have a common set of data files that you want to load for each script operation, you could simply create a script that loaded those files, then embed that script within your other scripts.

The following parameters are supported by the command:

- **FILENAME** - full path to script or workspace file to run
- **SKIP_UNLOAD_ALL** - specifies whether or not the first UNLOAD_ALL or UNLOAD_LAYER command in the script file being run should be skipped. This is useful for embedding workspace files which typically unload everything to start with. Use SKIP_UNLOAD_ALL=**YES** to skip the first UNLOAD_ALL or UNLOAD_LAYER command.

RUN_COMMAND

The RUN_COMMAND command allows you to execute any program on Windows with a set of parameters. The following parameters are supported by the command.

> This command may be disabled from within the application interface in the Configuration dialog in the General Advanced section for security. If it has been disabled, any scripts with RUN_COMMAND will not execute the command and will produce an error. It can not be re-enabled via scripting.

- **COMMAND_LINE** - full command line to run. If you need to use quotation marks in your command line, instead use apostrophes (i.e. ' rather than ") and they will be converted to quotes prior to running the command. Note to run a DOS shell command like 'mkdir', you will need to run it via cmd.exe, like COMMAND_LINE="cmd /C mkdir 'C:\temp\export test\new folder'" (note use of apostrophes for quotes).
- **PRESERVE_APOSTROPHES** - indicates whether or not apostrophes will be converted to double-quotes in the command string. Use PRESERVE_APOSTROPHES=**YES** to keep the apostrophes. The default is NO, so if you don't specify this parameter, the apostrophes in the command string will be converted to double-quote symbols.
- **WAIT_FOR_COMPLETE** - specifies whether or not the script should wait for your command line run to complete prior to continuing. The default is to wait for the command to complete (i.e. WAIT_FOR_COMPLETE=YES). If you just want the command line to run and then immediately let the script continue processing, use WAIT_FOR_COMPLETE=NO.
- **CAPTURE_RESULT** - specifies the name of a variable where the result of the program will be stored. The variable does not have to be created via a DEFINE_VAR command before it is used here, although it is OK if it is. The result will only be stored if RUN_COMMAND waits for the program to complete (see WAIT_FOR_COMPLETE).
- **HIDE_WINDOW** - specifies that any window launched by the command (like a command window) will initially be hidden. Add HIDE_WINDOW=**YES** to hide the window.

SAMPLES

Here is a sample that runs another instance of Global Mapper and loads a file:

```
RUN_COMMAND COMMAND_LINE="'c:\program files (x86)\GlobalMapper16\global_mapper.exe'
'c:\temp\export test\blue_springs.opt'" WAIT_FOR_COMPLETE=NO
```

Here is a sample that calls another .exe and stores the return code of the .exe to the variable RESULT:

```
RUN_COMMAND COMMAND_LINE="'c:\temp\test1.exe'" CAPTURE_RESULT="RESULT"
```

PLAY_SOUND

The PLAY_SOUND command plays either the information sound for the system or a specified sound file. This can be useful if you want audible confirmation when a script completes. The following parameters are supported by the command.

- **FILENAME** - full path to sound file (like .wav) to play. If not specified the information beep will play.

FORCE_EXIT

The FORCE_EXIT command aborts the script and optionally immediately shuts down Global Mapper without going through the normal shut-down process. This is useful if you are running a Global Mapper script via a CreateProcess call and the Global Mapper process is not returning when the script completes, or if you need a particular return code provided. The following parameters are supported by this command:

- **CLOSE_APP** - specifies whether or not the Global Mapper application should be closed in addition to stopping the script. By default this is enabled, so add CLOSE_APP=**NO** to only abort the script and not the entire app.
- **RETURN_CODE** - specifies the numeric return code to use. If not provided 0 is returned if the script did not encounter any errors or 1 if there were errors encountered.

LOG_MESSAGE

The LOG_MESSAGE command writes a string to the status window and any active log file. You can use the SET_LOG_FILE command to set the log file to save message to. The USER_FILENAME parameter of that command allows you to have LOG_MESSAGE text written to a different file than default script messages. You can include variables in the command string if you want to log their values. Everything on the line after the LOG_MESSAGE will be written. For example if you have a variable named WATER_LEVEL_FT, you could log its value and a timestamp at the front as follows:

```
LOG_MESSAGE %TIMESTAMP%: The current value of WATER_LEVEL_FT is %WATER_LEVEL_FT%
```

If you would like to log messages to the command line (if running a script passed on the command line), make sure to include LOG_TO_COMMAND_PROMPT=YES in the GLOBAL_MAPPER_SCRIPT header line at the start of the script.

Other built-in variables (see DEFINE_VAR) allow you to log the elapsed time (in seconds) for a script. For example you can log the time for an import and export and total script time using the following:

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
LOG_MESSAGE Script <%SCRIPT_FILENAME%> started at %DATE% %TIME%
IMPORT FILENAME="test.tif"
LOG_MESSAGE Import took %TIME_SINCE_LAST_LOG%
EXPORT_RASTER TYPE=GEOTIFF FILENAME="output.tif"
LOG_MESSAGE Export took %TIME_SINCE_LAST_LOG%
LOG_MESSAGE Total Script Run Time: %TIME_SINCE_START%
```

SET_LOG_FILE

The SET_LOG_FILE command sets the name of the file to log status, warning, and error messages to. If the log file specified already exists, the messages will be appended to the end of it. The following parameters are supported by the command.

- **FILENAME** - full path to log file to write messages to
- **USER_FILENAME** - full path to log file to write messages from the LOG_MESSAGE command. Use this if you want your own messages to go to a different file than any automatically generated script messages. Pass in with an empty value to reset the LOG_MESSAGE messages to go back to the shared file.

Conditional Execution in Global Mapper Scripts

Global Mapper scripts can incorporate conditional logic through the use of the IF, ELSE_IF, ELSE, and END_IF commands. At a minimum, when incorporating conditional logic into a script, the user must use an IF and an END_IF command, as in the following example:

```
IF COMPARE_STR="%VAR1%=val1"
// Additional script commands to be run when the condition is true.
END_IF
```

When the script containing the sample is run, if the value of the variable %VAR1% is "val1", the statements following the IF and before the END_IF will be executed. If the value of variable %VAR1% is something other than "val1", then the statements following the IF will be skipped, and script processing will continue with the first statement after the END_IF. If the user wants to run a specific set of commands in the case where the condition specified on the IF is not true, then he can use the ELSE command:

```
IF COMPARE_STR="%VAR1%=val1"
// Script commands to be run when the IF condition is true.
ELSE
// Script commands to be run when the IF condition is false.
END_IF
```

Now, if the value of the variable %VAR1% is "val1", the statements following the IF and before the ELSE will be executed, and the commands after the ELSE and before the END_IF will be skipped. If the value of variable %VAR1% is something other than "val1", then the statements following the IF and preceding the ELSE will be skipped, and the commands after the ELSE and before the END_IF will be run.

If the user has several conditions that need to be tested, only one of which can be true, then the ELSE_IF command can be used:

```
IF COMPARE_STR="%VAR1%=val1"
// Script commands to be run when the IF condition is true.
ELSE_IF COMPARE_STR="%VAR1%=val2"
// Script commands to be run when the ELSE_IF condition is true.
ELSE
// Script commands to be run when the all other conditions are false.
END_IF
```

The commands following the IF will be handled as described above, but now there is a second condition being tested. When the value of variable %VAR1% is "val2", then the commands after the ELSE_IF and before the ELSE will be run. In the case where there are multiple IF/ELSE_IF conditions, the commands after the ELSE will be run when all of the other conditions are false.

Logical Condition Commands

The IF and the ELSE_IF commands require a COMPARE_STR parameter to specify the condition to be tested.

The user can specify multiple COMPARE_STR parameters, all of which must be true to result in the subsequent commands being executed (logical AND). Use the COMPARE_OP="ANY" parameter to specify that the subsequent commands should be run if any one of the conditions is true (logical OR).

IF

If the comparison condition is true, perform the subsequent commands. The if statement is a code block that must be closed with an END_IF.

- **COMPARE_STR** - The COMPARE_STR must consist of <value><operation><value>, where either value can be a constant or a variable name (enclosed in "%"). Both values must be specified. The operation is also required, and can be one of:
    - **"="** - Equals
    - **"!="** - Not Equals
    - **"<"** - Less Than
    - **"<="** - Less Than or Equal To
    - **">"** - Greater Than
    - **">="** - Greater Than or Equal To
- **COMPARE_NUM** - By default, the comparison will be a case-insensitive string comparison. If you want to perform a numeric comparison, specify the COMPARE_NUM=**"YES"** parameter.
- **COMPARE_OP** - If multiple compare strings (COMPARE_STR) are defined, specify the handling option for the if statement to be considered true.
    - **ANY** - With multiple comparisons, the result is true if one or more of the individual conditions is true (logical OR)
    - **ALL** - This is the default. All comparisons must be true for the command following the if statement to be executed.

ELSE_IF

Second comparison condition can be tested, and if it is true, the subsequent command will be executed. The available parameters for are the same as IF, see above.

ELSE

All other cases that do not return true for the IF or ELSE_IF conditions will have the commands following the else statement executed.

END_IF

IF commands can be nested, so the block of commands following an IF, ELSE_IF, or ELSE command can contain another IF command:

```
IF COMPARE_STR="%VAR1%=val1"
IF COMPARE_STR="%VAR2%>10"
// Script commands to be run when the IF condition is true.
END_IF
ELSE
// Script commands to be run when the IF condition is false.
END_IF
```

Looping Operations

DIR_LOOP_START

The DIR_LOOP_START command begins a loop of commands over all of the folders within a directory (and option-ally its subdirectories) that match one or more filename masks. This is a powerful feature allowing you to do things like easily batch convert a collection of files or perform any other supported operation over a collection of files. You end a loop over the files in a folder using the DIR_LOOP_END command. Note that it is also possible to nest loops.

Built-in Variables

For any commands found within a DIR_LOOP_START...DIR_LOOP_END pair defining a loop, the following special character sequences can be used anywhere (examples of what the values will be based on a current filename of 'C:\path\to\my\data\my_file.dem' are listed):

- *%DIR%* - full path to current file (value is 'C:\path\to\my\data\')
- *%FNAME_W_DIR%* - full path and filename of current file (value is 'C:\path\to\my\data\my_file.dem')
- *%FNAME%* - filename of current file (value is 'my_file.dem')
- *%FNAME_WO_EXT%* - filename of current file without extension (value is 'my_file')
- *%PARENT_DIR%* - name of parent directory of file (value is 'data')
- *%PARENT_DIRN%* - name of some level of parent directory, where 'N' is level. For example, %PARENT_DIR1% value is 'my' and %PARENT_DIR2% is 'to'.
- *%RECURSE_FOLDER%* - folder recursed into beyond original search folder. So if current filename was 'C:\path\to\my\data\sub\folder\my_file.dem' the value of this would be 'sub\folder\'. Use this to rebuild a directory structure elsewhere when recursing.

For a sample of the DIR_LOOP_START command in use, see the example at the bottom of this reference.

The following parameters are used by the DIR_LOOP_START command.

- **DIRECTORY** - specifies the directory to search for files in. If you leave this blank, the operation will be based in the current folder.
- **FILENAME_MASKS** - space-separated list of filename masks to match on. If no value is provided then all files will be used. If you provide ".." as the mask, you will enter the loop once for each folder that is matched, allowing you to perform one operation per folder on an enter directory tree. In addition to indi-vidual masks the following special values are also supported:
  - **COMMON_ALL** - a filter with all Commonly Supported Formats
  - **COMMON_ELEV** - a filter with all Commonly Supported Elevation Grid Formats
  - **COMMON_RASTER** - a filter with all Commonly Supported Raster Formats
  - **COMMON_VECTOR** - a filter with all Commonly Supported Vector Formats

- **FILENAME_MASKS_EXCLUDE** - space-separated list of filename masks to exclude from matching. Any files that match the FILENAME_MASKS but also match a FILENAME_MASKS_EXCLUDE mask will not be looped over.
- **RECURSE_DIR** - specifies whether the loop operation will search subdirectories of the specified directory as well as the current one. Use RECURSE_DIR=YES to enable. The default value is to NOT search sub-directories.
- **LIST_FILENAME** - specifies an explicit list of files to iterate over in order. The value can either refer to a previously defined inline DEFINE_TEXT_FILE or a text file on disk. Each line should contain the filename to load.
- **INDEX_VAR** - specifies the name of a variable to initialize with the current loop index. For example, specify INDEX_VAR="FILE_IDX", then you can use %FILE_IDX% inside the loop. It will have values starting at 0 and incrementing by 1.

DIR_LOOP_END

The DIR_LOOP_END command ends a loop of commands over all of the folders within a directory. See the DIR_LOOP_START command for details.

VAR_LOOP_START

The VAR_LOOP_START command begins a loop of commands over a range of numeric values or through a sequence of characters. This can be used as a simple counter or for more powerful things like custom gridding using coordinate values and naming exported files using the coordinates. Note that it is possible to nest loops and use different variable names for each loop to build complex filenames.

For any commands found within a VAR_LOOP_START...VAR_LOOP_END pair defining a loop, the current value of the loop variable will be available as a variable name. By default this will be %COUNTER%, but you can use the VAR_NAME parameter (see below) to make it whatever name that you want. By default some generic numeric formatting will be provided (i.e. whole numbers won't have a decimal or any leading 0's), but you can also provide custom formatting for the numeric value as a C-style format string like you would pass to a print command using the VAL_FORMAT parameter (see below for details).

The following parameters are used by the VAR_LOOP_START command.

- **VAR_NAME** - specifies the name of the variable that will be used to store the current loop value. By default this will be %COUNTER%, but you can use anything you want. See the example below for usage.
- **VAL_START** - specifies the value to start the loop out. This would be something like 1 for just a simple counter loop, but can be any number, or a single letter, like A.
- **VAL_STOP** - specifies the value to stop the loop out. When the current loop value goes past this value the loop will stop, but it will run at this value. So to do a loop from 1 to 10, including 10, use VAL_START=1, VAL_STOP=10, and VAL_STEP=1. To loop through the letters A through J, use VAL_START=A, VAL_STOP=J, and VAL_STEP=1.
- **VAL_STEP** - specifies the value to increment the loop variable by each time the commands are run through. If you don't provide this it will increment by 1 if the VAL_STOP is greater than VAL_START and -1 if they are reversed.
- **VAL_FORMAT** - specifies a C-style printf format string for formatting the numeric loop variable as a string. For example to format as a 3-digit number with 0's filling in for values less than 100, use VAL_FORMAT="%03d". If you provide a custom format, it should always include exactly one % and end with a 'd' (for integer values) or a 'f' (for floating point). If you don't provide a format string a good default numeric represntation will be used.

VAR_LOOP_END

The VAR_LOOP_END command ends a loop of commands over a range of numeric values. See the VAR_LOOP_START command for details.

SAMPLE

Here is a simple example for looping over some rows and columns:

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
// Loop over rows 1-10 with leading zeroes in the format
VAR_LOOP_START VAL_START=1 VAL_STOP=10 VAL_STEP=1 VAL_FORMAT="%02d" VAR_NAME="%ROW%"
    // Loop over colums 5-15 in this row, use default formatting
        VAR_LOOP_START VAL_START=5 VAL_STOP=15 VAL_STEP=1 VAR_NAME="%COL%"
         // Import a file with the row and column in the filename
         IMPORT FILENAME="c:\path_to_file\base_filename_%ROW%_%COL%.jpg"
        VAR_LOOP_END
VAR_LOOP_END
```

LAYER_LOOP_START

The LAYER_LOOP_START command begins a loop of commands over load layers. You can loop over all layers or just those matching a particular filename mask. You end a loop over the files in a folder using the LAYER_LOOP_END command. Note that it is also possible to nest loops.

Built-in Variables

For any commands found within a LAYER_LOOP_START...LAYER_LOOP_END pair defining a loop, the following special character sequences (the LAYER part can be changed using the VAR_NAME_PREFIX parameter) can be used anywhere (examples of what the values will be based on a current layer filename of 'C:\data\my_file.dem' are listed):

- *%LAYER_DIR%* - full path to current file (value is 'C:\data\')
- *%LAYER_FNAME_W_DIR%* - full path and filename of current file (value is 'C:\data\my_file.dem')
- *%LAYER_FNAME%* - filename of current file (value is 'my_file.dem')
- *%LAYER_FNAME_WO_EXT%* - filename of current file without extension (value is 'my_file')
- *%LAYER_PARENT_DIR%* - name of parent directory of file (value is 'data')
- *%LAYER_DESC%* - description of current layer

The following parameters are used by the LAYER_LOOP_START command.

- **FILENAME** - filename or description of layer(s) to loop over. This can include * and ? wildcard characters. If you leave the FILENAME parameter off then all loaded layers will be looped over, which is the same behavior as using FILENAME="*". If you specify a blank FILENAME parameter then you will loop over all layers not based on a file. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value **'USER CREATED FEATURES'** to have the 'User Created Features' layers looped over or **'SELECTED LAYERS'** to have any layers selected in the Control Center looped over.
- **VAR_NAME_PREFIX** - prefix to use for the variable names in the loop (useful in nested loops). For example if you provided VAR_NAME_PREFIX="HIDE", then you would use %HIDE_FNAME% rather than %LAYER_FNAME% inside that loop. If you don't provide a value then the default of LAYER is used.

LAYER_LOOP_END

The LAYER_LOOP_END command ends a loop of commands over loaded layers. See the LAYER_LOOP_START command for details.

SAMPLE

```
GLOBAL_MAPPER_SCRIPT VERSION="1.00"
// Hide all layers
LAYER_LOOP_START FILENAME="*" VAR_NAME_PREFIX="HIDE"
    SET_LAYER_OPTIONS FILENAME="%HIDE_FNAME_W_DIR%" HIDDEN=YES
LAYER_LOOP_END
// Loop over the loaded layers, doing a separate export for each
LAYER_LOOP_START FILENAME="*"
```

```
    // Enable the current layer since we hid it above
    SET_LAYER_OPTIONS FILENAME="%LAYER_FNAME_W_DIR%" HIDDEN=NO

    // Export
    EXPORT_RASTER FILENAME="%LAYER_DIR%%LAYER_FNAME_WO_EXT%_loop.tif" FORMAT=GEOTIFF

    // Disable the current layer so it won't be involved in other operations
    SET_LAYER_OPTIONS FILENAME="%LAYER_FNAME_W_DIR%" HIDDEN=YES
LAYER_LOOP_END
// Unhide all layers
LAYER_LOOP_START FILENAME="*" VAR_NAME_PREFIX="HIDE"
    SET_LAYER_OPTIONS FILENAME="%HIDE_FNAME_W_DIR%" HIDDEN=NO
LAYER_LOOP_END
```

Define data

DEFINE_SHAPE

The DEFINE_SHAPE command allows a multi-point shape (like a polygon) to be associated with a name. The shape name can then be used in later commands for things like cropping and feathering to polygonal boundaries.

The DEFINE_SHAPE command consists of a single command line followed by a series of lines describing the series of XY coordinate pairs that define the shape. Each line should have a single coordinate value with the X and Y coordinates separated by a comma. If you need your shape to contain multiple boundaries, insert **BREAK_SHAPE** on a line to stop the current sequence and start a new one.

The DEFINE_SHAPE command is terminated with a single line containing only the text **END_DEFINE_SHAPE**.

The following parameters are required by the DEFINE_SHAPE command.

- **SHAPE_NAME** - specifies the name to associate with the shape

SAMPLE

Here is an example of a DEFINE_SHAPE command used to define a feather polygon with a name of 'FEATHER_POLY'

```
DEFINE_SHAPE SHAPE_NAME="FEATHER_POLY"
377493.234,4323974.016
375343.359,4318676.109
381101.953,4314414.750
387014.109,4317178.875
386975.719,4322400.000
381869.766,4324588.266
377493.234,4323974.016
END_DEFINE_SHAPE
```

Here is an example of a DEFINE_SHAPE command used to define a feather polygon with 2 separate boundaries with a name of 'FEATHER_POLY'

```
DEFINE_SHAPE SHAPE_NAME="FEATHER_POLY"
377493.234,4323974.016
375343.359,4318676.109
381101.953,4314414.750
387014.109,4317178.875
377493.234,4323974.016
```

```
 BREAK_SHAPE
386975.719,4322400.000
381869.766,4324588.266
377493.234,4323974.016
386975.719,4322400.000
END_DEFINE_SHAPE
```

DEFINE_TEXT_FILE

The DEFINE_TEXT_FILE command allows an ASCII text file containing the definition of vector features to be associated with a name. You can then use the associated name in the FILENAME parameter of the IMPORT_ASCII command to load the file contents as if they were an external file. This is a powerful command allowing you to embed the definition of vector features directly within a script.

The associated name can also be used with the DIR_LOOP_START command where the definition includes a list of filepaths.

The DEFINE_TEXT_FILE command consists of a single command line followed by a series of lines with the contents of the text "file". The DEFINE_TEXT_FILE command is terminated with a single line containing only the text END_DEFINE_TEXT_FILE.

The following parameters are required by the DEFINE_TEXT_FILE command.

- **FILENAME** - specifies the name to associate with this text file contents. Provide this value in the FILENAME parameter for the IMPORT_ASCII command.

SAMPLE

Here is an example of a DEFINE_TEXT_FILE command used to define a couple of feature shapes.

```
DEFINE_TEXT_FILE FILENAME="Test Features"
GM_TYPE=Lake, < 0.5 sq. mi.
DESCRIPTION=LAKE OR POND
BORDER_COLOR=RGB(0,0,0)
BORDER_STYLE=Null
FILL_COLOR=RGB(0,0,211)
FILL_STYLE=Solid Fill
CLOSED=YES
LABEL_POS=382285.8,4331317.0
DLGMAJ_0=50
DLGMIN_0=421
382277.3,4331322.5
382297.4,4331322.5
382297.4,4331322.5
382290.6,4331312.0
382277.7,4331311.0
382277.7,4331311.0
382277.3,4331322.5
GM_TYPE=Minor River
DESCRIPTION=STREAM
BORDER_COLOR=RGB(0,0,0)
BORDER_STYLE=Null
FILL_COLOR=RGB(0,0,211)
FILL_STYLE=Solid Fill
CLOSED=YES
LABEL_POS=381081.4,4319924.0
DLGMAJ_0=50
DLGMIN_0=412
381269.3,4320021.0
381288.4,4320030.0
```

```
381298.8,4320037.0
381330.0,4320081.5
381359.7,4320117.0
381380.0,4320130.0
381397.2,4320136.0
381419.2,4320136.5
381460.5,4320125.5
381517.1,4320120.5
381554.9,4320121.5
381627.6,4320133.0
381665.4,4320135.5
381684.3,4320131.5
381707.9,4320122.0
381725.6,4320118.5
381733.4,4320112.5
END_DEFINE_TEXT_FILE
```

DEFINE_VAR

The DEFINE_VAR command allows you to define a variable and an associated value. You can then use the defined variable name later wrapped in percent signs to have the defined value replaced in the script. This is useful for things like defining a path or something at the top of a script that you can easily change in just one place later. You can also pass variables on the command line for truly power batch-mode operation.

Built-in Variables

There are several built-in variable names that you can use to easily insert things like the current date and time. The following variable strings can be used without having to define them:

- *%TIMESTAMP%* - inserts current date and time in system format
- *%TIMESTAMP_MS%* - inserts current date and time to millisecond resolution in the format 'YYYYMMDD_ HHMMSSsss'
- *%DATE%* - inserts current date in system format
- *%TIME%* - inserts current time in system format
- *%TIME_SINCE_START%* - inserts the number of seconds since the script starting running
- *%TIME_SINCE_LAST_LOG%* - inserts the number of seconds since the last use of this variable
- *%SCRIPT_FILENAME%* - inserts the full path and filename of the running script
- *%SCRIPT_FOLDER%* - inserts the full path of the running script/workspace file. This will include a trailing slash. So a script 'C:\path\my_script.gms' would get 'C:\path\' inserted
- *%GM_MAJOR_VER%* - inserts the major version of Global Mapper. For example, any Global Mapper v17 build would insert '17'
- *%GM_FULL_VER_W_DATE%* - inserts the full version of Global Mapper, including the build date. For v17.0.0 built on Sep 22, 2015, this would be 'v17.0.0 (b092215)'
- *%GM_FULL_VER_NO_DATE%* - inserts the full version of Global Mapper. For v17.0.0 this would be 'v17.0.0'

The following parameters are required by the DEFINE_VAR command.

- **NAME** - specifies the variable name
- **VALUE** - specifies the variable value
- **FORMULA** - specifies a formula used to calculate the variable's value. The formula is specified similarly to those used in the CALC_ATTR_FORMULA command, except that feature attributes cannot be specified, but instead, other defined variables, including the predefined variables, may be used. See the formula calculator reference documentation for details. Note that if FORMULA is specified, then all other parameters

except for NAME will be ignored.

To use other variables, either user-created or predefined, the delimiting '%' characters should not be used, since variable replacement in the scripting engine is performed in script commands before inter-pretation, including in the FORMULA parameter, an invalid formula will likely result. For example, use:

```
DEFINE_VAR NAME="TIMEDATE" FORMULA="match(TIME, '\d+:\d+')"
```

rather than

```
DEFINE_VAR NAME="TIMEDATE" FORMULA="match(%TIME%, '\d+:\d+')"
```

Matching of variable names is performed via a case-sensitive match. If a variable is used in formula, but it doesn't exist, then it will be replaced by an empty string.

- **REPLACE_STR** - specifies a text value to replace inside the value with something else. This is typically used inside a DIR_LOOP_START...DIR_LOOP_END loop where the VALUE contains other variables. The format is REPLACE_STR="old_value=new_value". See example below.

- **PROMPT** - specifies that the user should be prompted to enter the value for the variable rather than spe-cifying it with the VALUE parameter. Very useful for developing interactive scripts. The following values are recognized:
    - **YES** - Display prompt with OK and Cancel buttons and a box to enter the value.
    - **YES_NO** - Display prompt with Yes and No buttons. The value is set to YES or NO depending on what is selected.
    - **YES_NO_CANCEL** - Display prompt with Yes and No buttons. The value is set to YES or NO depend-ing on what is selected. If Cancel is pressed the operation is cancelled.
    - **OK** - Display an information message with an OK button.

- **FILE** - Prompts the user for a filename. The PROMPT text will be the title of the file open dialog. The VALUE (if any) will be the default filename selection (can be full path to provide a default folder too). You can use the VALUE parameter to provide several defaults, including default folder, default filename, and/or default file extension. If you just want to provide a default folder, use VALUE="C:\PATH_HERE\." where the file-name is just a dot.
    - **DIR** - Prompts the user for a directory/ folder. The PROMPT text will be the title of the folder selection dialog. The VALUE (if any) will be the default folder selection.

- **PROMPT_TEXT** - specifies the text to show if a PROMPT parameter is provided

- **ABORT_ON_CANCEL** - specifies that if a prompt is cancelled (like for a file) that the entire script should be aborted. Defaults to YES if a cancellable prompt is provided. Use ABORT_ON_CANCEL=NO to not cancel the whole script on cancel of prompt.

- **FILE_MUST_EXIST** - specifies that if PROMPT=FILE is specified, the selected file can be a new one and doesn't have to already exists.
    - FILE_MUST_EXIST=**YES** will pop up a File Open dialog, and the user must select an existing file.
    - FILE_MUST_EXIST=**NO** will pop up an File Save dialog, and the user can choose an existing file or a new one. This is the current default.

- **VALUE_ATTR** - specifies the variable value should come from the first feature in the specified layer that has a non-empty value for the specified attribute. You should provide a FILENAME parameter to indicate which layer(s) to check for the attribute, or none to check all of them. See special Attribute Name para-meter details.

- **FILENAME** - filename of the layer to get the attribute value from if you use VALUE_ATTR. If an empty value is passed in, all loaded vector layers will be checked. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value *'USER CREATED FEATURES'* to have the 'User Created Features' layer updated or 'SELECTED LAYERS' to have any layers selected in the Control Center updated.

- **FILENAME_PIECE** - specifies that only a piece of the specified value should be used. Used if the value spe-cifies a full path and filename and you want to define the variable to just a piece of that. So you might use one DEFINE_VAR with a PROMPT=FILE to select a filename, then another DEFINE_VAR with that previous variable value as the value and FILENAME_PIECE added requesting one of the pieces listed below (samples based on 'C:\path\to\my\data\my_file.dem'):

- **DIR** - full path to current file (value is 'C:\path\to\my\data\')
- **FNAME** - filename of current file (value is 'my_file.dem')
- **FNAME_WO_EXT** - filename of current file without extension (value is 'my_file')
- **PARENT_DIR** - name of parent directory of file (value is 'data')
- **PARENT_DIR*N*** - name of some level of parent directory, where 'N' is level. For example, PARENT_DIR1 value is 'my' and %PARENT_DIR2% is 'to'.

- **VALUE_TABLE** - Provides the name of the table to be queried. See DEFINE_VAR_TABLE command.

- **VALUE_COLUMN** - Provides the name of the column that contains the data used to set the value of the parameter. See DEFINE_VAR_TABLE command.

- **COMPARE_STR** - Indicates which row to use for setting the value from a value table. See DEFINE_VAR_TABLE command.

- **METADATA_LAYER** - identifies the layer from which the metadata will be copied. This is the same value that would be passed to the FILENAME parameter on the IMPORT command used to load the layer.

- **METADATA_ATTR** - This is the string used to identify the metadata from which the value will be copied. The complete list of metadata attribute names for a layer can be seen by clicking the Metadata... button on the Control Center, or by running the EXPORT_METADATA command and looking at the result file. Some example metadata attributes are "**UPPER LEFT X**", "**AREA COUNT**", "**PROJ_DESC**", etc.

SAMPLE

Here is an example of a DEFINE_VAR command used to define a directory path for later use and then its subsequent use:

```
DEFINE_VAR NAME="DATA_DIR" VALUE="c:\temp\export test"
IMPORT FILENAME="%DATA_DIR%\blue_springs.opt"
```

For example you could use the following inside a directory loop to change the output path:

```
DEFINE_VAR NAME="OUT_FNAME" VALUE="%FNAME_W_DIR%" REPLACE_STR="OLD_PATH\=NEW_PATH\SUB_FOLDER\"
```

Example to define a variable based on the F_CODE attribute of a loaded layer and then use that in the layer description of the layer.

```
DEFINE_VAR NAME="LAYER_F_CODE" VALUE_ATTR="F_CODE" \
FILENAME="c:\path_to_layer\my_data.shp"
SET_LAYER_OPTIONS FILENAME="c:\path_to_layer\my_data.shp" \
LAYER_DESC="%LAYER_F_CODE%"
```

Example to prompt the user for a folder and then a .zip file in that folder (you could have just prompted for the file all at once, but this is for demonstration), then loads it.

```
// Prompt for folder to load
DEFINE_VAR NAME="DIR_TO_LOAD" PROMPT="DIR" VALUE="d:\temp\export test\" \
ABORT_ON_CANCEL=NO
// Prompt for .zip file to load in folder. Cancel if nothing selected.
DEFINE_VAR NAME="FILE_TO_LOAD" PROMPT="FILE" VALUE="%DIR_TO_LOAD%.zip" ABORT_ON_CANCEL=YES
// Load the file.
IMPORT FILENAME="%FILE_TO_LOAD%"
```

Example to prompt the user for a filename, then define new variables that hold pieces of the selected filename.

```
// Prompt use for file. Script aborts on cancel
DEFINE_VAR NAME="FULL_FNAME" PROMPT=FILE ABORT_ON_CANCEL=YES
// Define a variable with just the filename, another with parent folder
DEFINE_VAR NAME="FNAME_ONLY" VALUE="%FULL_FNAME%" FILENAME_PIECE="FNAME_WO_EXT"
```

```
DEFINE_VAR NAME="FNAME_PARENT_DIR" VALUE="%FULL_FNAME%" FILENAME_PIECE="PARENT_DIR"
DEFINE_VAR NAME="FNAME_PARENT_DIR1" VALUE="%FULL_FNAME%" FILENAME_PIECE="PARENT_DIR1"
```

DEFINE_VAR_TABLE

The command can be used to set up a look-up table in the script. Once the table is set up, you can use a DEFINE_ VAR command to set up a script variable by looking up a value in the table. The data that makes up the table can be specified inline, or read from a CSV file. GM assumes that the first line of the data contains the column names for the table. This is true regardless of whether the data is inline or in a file.

The following parameters are required by the DEFINE_VAR_TABLE command.

- **NAME** - Defines the name of the table
- **FILENAME** - (optional) Provides the full path to the CSV file that contains the table data. If both a file name and inline data are provided, only the data from the file is used; the inline data will be ignored.

END_VAR_TABLE

- Indicates the end of the DEFINE_VAR_TABLE command. This is always required, whether the data is specified inline or in a file.

SAMPLES

Example (data specified inline):

```
DEFINE_VAR_TABLE NAME="state_codes"
st_code,st_abb,st_name
06,ca,"california,xyz"
08,co,colorado
END_VAR_TABLE
DEFINE_VAR NAME="st_abb1" VALUE_TABLE="state_codes" VALUE_COLUMN="st_abb" \
COMPARE_STR="st_code=06"
DEFINE_VAR NAME="st_name1" VALUE_TABLE="state_codes" VALUE_COLUMN="st_name" \
COMPARE_STR="st_code=06"
```

Example (data in a file):

```
DEFINE_VAR_TABLE NAME="state_codes" \
FILENAME="C:\Temp\GlobalMapperWorkspaces\test_tabl e.csv"
END_VAR_TABLE
DEFINE_VAR NAME="st_abb5M" VALUE_TABLE="state_codes" VALUE_COLUMN="st_abb" \
COMPARE_STR="st_code=09"
DEFINE_VAR NAME="st_name5M" VALUE_TABLE="state_codes" VALUE_COLUMN="st_name" \
COMPARE_STR="st_code=09"
```

The example CSV file contains the following data:

> *st_code,st_abb,st_name*
>
> *06,ca,"california,xyz"*
>
> *08,co,colorado*
>
> *09,me,maine*

Display

DEFINE_PROJ

The DEFINE_PROJ command allows a projection (including datum) to be associated with a name. The projection name can then be used in later IMPORT, IMPORT_ARCHIVE, IMPORT_ASCII, and LOAD_PROJECTION commands to specify a projection as needed.

The DEFINE_PROJ command consists of a single command line followed by a series of lines describing the projection in the format of an ESRI PRJ file. The easiest way to determine the text for a projection is to setup a projection on the Projection tab of the Tools->Configuration and then use the Save to File button to create a new .prj file. Then just open the .prj file up in Notepad and copy the contents to the lines following the DEFINE_PROJ command line.

The DEFINE_PROJ command is terminated with a single line containing only the text **END_DEFINE_PROJ**.

For a sample of the DEFINE_PROJ command in use, load some data and then save a Global Mapper workspace file from the File->Save Workspace menu command. Open the resulting .gmw file in an editor and you can see how the DEFINE_PROJ command is used to define a view projection and the set it.

The following parameters are required by the DEFINE_PROJ command.

- **PROJ_NAME** - specifies the name to associate with the projection

LOAD_PROJECTION

The LOAD_PROJECTION command imports a projection from a PRJ file and makes it the current global projection. This projection will be used for all exports after this command until another LOAD_PROJECTION command is

encountered to change the global projection. The following parameters are supported by the command (you would just use one of the below):

- **PROJ** - special Projection Specification type of parameter that specifies the projection to use for the file. This will override any projection information stored in the file.
- **FILENAME** (DEPRECATED use PROJ) - full path to PRJ file to load the projection from
- **PROJ_NAME** (DEPRECATED use PROJ) - specifies the name of the projection to use. This name must have been defined with a prior DEFINE_PROJ command.
- **PROJ_EPSG_CODE** (DEPRECATED use PROJ) - specifies the numeric EPSG projection code that defines the projection. For example, use `PROJ_EPSG_CODE=26715` to define a UTM zone 15 projection with NAD27 as the datum and meters as the units.

Projection Specification Values

A projection specification parameter has a value that can specify a projection in the following ways:

- *PRJ Filename* - the value is the full path to a PRJ file that specifies the projection to use.
- *Defined Projection Name* - the value is the name assigned to a projection previously defined with the DEFINE_PROJ command.
- *EPSG Code* - the value is an EPSG code, either just the raw code, or a code with **EPSG:** in front of it, like `'EPSG:4326'`.
- *WKT PRJ String* - the value is a WKT projection string.
- *Zoned Projection Name* - the value is the name of a zoned projection system or of a predefined grid system with no parameters (i.e. British Grid, Swiss Grid, etc.). For zoned projections, the zone to use will be automatically selected for the center location of the loaded data. The datum of the current projection will be used as will the units of the current projection (or meters if the current projection uses degrees). For example, `PROJ="UTM"` will select a UTM projection with the appropriate zone for the center lat/lon of the loaded data. Here are some recognized zoned projection names (or just use the name from the projection dialog):
    - **UTM** - Universal Transverse Mercator (6-degree zones)
    - **GK3** - Gauss Krueger 3-degree zones
    - **GK6** - Gauss Krueger 6-degree zones
    - **MTM_CANADA** - MTM Eastern Canada
    - **AMG** - Australian Map Grid
    - **MGA** - Map Grid Australia
    - **3TM** - 3TM Alberta
    - **10TM** - 10TM Alberta

SAVE_PROJECTION

The SAVE_PROJECTION command saves the current global projection to a PRJ file. The following parameters are supported by the command.

- **FILENAME** - full path to PRJ file to save the projection to

DEFINE_SHADER

The DEFINE_SHADER command allows a custom elevation / slope shader to be defined to be used when rendering gridded elevation data. The shader will then be available for any other operations and later Global Mapper runs. If there is an existing custom shader with the same name it will be replaced.

The DEFINE_SHADER command consists of a single command line followed by a series of lines describing the series of elevation / slope and color pairs that define the shader. Each line should have a single elevation / slope value and a color value separated by a comma.

The DEFINE_SHADER command is terminated with a single line containing only the text **END_DEFINE_SHADER**.

The following parameters are required by the DEFINE_SHADER command.

- **SHADER_NAME** - specifies the name to associate with the shader
- **BLEND_COLORS** - specifies whether or not colors should smoothly blend between values. Use BLEND_ COLORS=**NO** to disable blending beween colors so you only get exactly the specified color.
- **STRETCH_TO_RANGE** - specifies whether or not the specified elevation values for the shader should stretch to the range of the loaded data. Use STRETCH_TO_RANGE=**YES** to enable.
- **SHADE_SLOPES** - specifies whether the elevation/slope values are slopes in degrees or elevation values. Use SHADE_SLOPES=**YES** to indicate the specified values are slopes.
- **SLOPES_PERCENT** - specifies whether the slope values that are specified. Add SLOPES_PERCENT=**YES** to indicate that slopes are in percent, otherwise they will be in degrees.
- **OVERWRITE_EXISTING** - specifies what to do if there is already a custom shader with the given name If OVERWRITE_EXISTING=**YES** is specified the existing custom shader will be overwritten. Otherwise it would be ignored.
- **SAVE_SHADER** - specifies whether or not the shader should be saved between runs via the custom_ shaders.txt file. This is enabled by default, so add SAVE_SHADER=**NO** to cause your shader to NOT be remembered.

SAMPLE

Here is an example of a DEFINE_SHADER command used to define a sample 3-color blended shader.

```
DEFINE_SHADER SHADER_NAME="Script Shader" BLEND_COLORS=YES STRETCH_TO_RANGE=YES SHADE_SLOPES=NO
0.0,RGB(255,0,0)
20.0,RGB(0,255,0)
50.0,RGB(0,0,255)
END_DEFINE_SHADER
```

DEFINE_LAYER_STYLE

The DEFINE_LAYER_STYLE command allows you to define a layer style for the area, line, or point features in a vector layer. You can then apply this with the IMPORT or SET_LAYER_OPTIONS command later.

The DEFINE_LAYER_STYLE command consists of a single command line followed by a series of lines with the contents of a .gm_layer_style file, like you would save from the Area Styles, Line Styles, or Point Styles tab of the Options dialog for a vector layer. You can also provide a .gm_layer_style in the FILENAME parameter to just reference an external file rather than embedding the layer style directly in the script file.

The DEFINE_LAYER_STYLE command is terminated with a single line containing only the text **END_DEFINE_ LAYER_STYLE**.

For a sample of the DEFINE_LAYER_STYLE command in use, load some data and set up some attribute-based or other custom styling on the Styles tabs of the Options dialog, then save a Global Mapper workspace file from the File->Save Workspace menu command. Open the resulting .gmw file in an editor and you can see how the DEFINE_LAYER_STYLE command is used to define a layer style and then use it in the IMPORT command.

The following parameters are required by the DEFINE_LAYER_STYLE command.

- **NAME** - specifies the name to associate with the style. Use this value in an IMPORT command with the AREA_STYLE_NAME, LINE_STYLE_NAME, or POINT_STYLE_NAME parameters.
- **TYPE** - type of style being specified, allowed values are **AREA**, **LINE**, and **POINT**.
- **FILENAME** - specifies the name of a .gm_layer_style file to use for the style rather than embedding it.

LOAD_STYLE_FILE

The LOAD_STYLE_FILE command load a Global Mapper Style (.gm_style) file containing style definitions for a list of types. You can optionally choose to have any types specified in the style file that aren't present in the running instance of Global Mapper to be added, providing a script way to add new custom types. The following parameters are supported by the command:

- **FILENAME** - full path to style (.gm_style) file to load
- **ADD_UNKNOWN_TYPES** - specifies that any types found in the style file that aren't present will be added as custom types. Use ADD_UNKNOWN_TYPES=**NO** to disable adding missing types, or ADD_UNKNOWN_TYPES=**YES** to enable it (this is the default).

SET_OPT

The SET_OPT command provides a place to set general options that aren't layer-specific, like the Position Display Format (see POS_DISP_FORMAT below). Use the SET_VERT_DISP_OPTS command for options related to the display of terrain data. The functionality that used to be in the LOAD_TYPE_FILTER command is now found here as well. The following parameters are supported by the command:

- **POS_DISP_FORMAT** - specifies the Position Display Format to use. This is the same as the setting on the General tab of the Configuration dialog. This setting is used for things like CSV export with EXPORT_VECTOR and with the ADD_COORD_ATTRS parameter to the EDIT_VECTOR command. The following values are supported:
  - **DECIMAL** - Lat/lon Degrees formatted as DD.DDDDDD N/S/E/W
  - **DECIMAL_PLAIN** - Lat/lon Degrees formatted the same as DECIMAL except for no hemisphere letter at the end
  - **DMS** - Lat/lon Degrees formatted as DD MM SS.SSSS N/S/E/W
  - **DM** - Lat/lon Degrees formatted as DD MM.MMMM N/S/E/W
  - **MGRS** - Position formatted as MGRS/USNG
- **AREA_UNITS** - specifies the units to use when storing area measurements. The following values are supported:
  - **ACRES**
  - **HECTARES**
  - **SQUARE FEET**
  - **SQUARE KILOMETERS**
  - **SQUARE METERS**
  - **SQUARE MILES**
- **DISTANCE_UNITS** - specifies the units to use when storing linear distance measurements. The following values are supported:
  - **METRIC** - meters for shorter distances, kilometers for longer
  - **STATUTE** - feet for shorter distances, miles for longer
  - **YARDS** - yards for shorter distances, kilometers for longer
  - **NAUTICAL** - feet for shorter distances, nautical miles for longer
  - **CHAINS** - chains for shorter distances, miles for longer
- **MEASURE_UNIT_TYPE** - specifies how to handle measurement values of different sizes
  - **AUTO** - automatically use base units for smaller measurements and large units for long (i.e. meters for shorter distances, kilometers for longer)
  - **BASE** - always use base units, regardless of size. For example, always use meters for distance. Use this if you want to numerically compare values
  - **LARGE** - always use large units, regardless of size. For example, always use kilometers for distance. Use this if you want to numerically compare values
- **DRAW_AREAS** - specifies whether or not area features are drawn. Use DRAW_AREAS=**YES** to enable.
- **DRAW_LINES** - specifies whether or not line features are drawn. Use DRAW_LINES=**YES** to enable.
- **DRAW_POINTS** - specifies whether or not point features are drawn. Use DRAW_POINTS=**YES** to enable.
- **DRAW_LABELS** - specifies whether or not feature labels are drawn. Use DRAW_LABELS=**YES** to enable.
- **DRAW_ORDER** - specifies the order in which layers are drawn, in particular vector layers. This corresponds to the Vector Layer Order During Draw setting on the Vectory Display tab of the Configuration dialog.
  - **BY_TYPE** - Layers are ordered by type. Any raster/elevation grid layers are drawn first in whatever order they are organized in the Control Center (default is load order). After that, vector features are sorted together with areas drawing first (sorted by type/style/layer within areas), then line, and finally point features.
  - **BY_LAYER** - Layers are drawn in the order they are shown in the Control Center (i.e. default is load order). Within a single vector layer, areas are drawn first, then lines, and finally points.
- **MISC_OPT** - specifies an advanced option to set the value of. There are a number of named options listed below. In addition to those, any numeric value from the GM_MiscOpt_t32 type defined in the GlobalMapperInterface.h of the SDK are available to be passed in. Use the MISC_OPT_VALUE to set the value for

the option.

- **MAINTAIN_EXPORT_BOUNDS** - controls whether or not when providing a bounding box and resolution for an export, the bounds will be exactly maintained and the sample spacing slightly shrunk if necessary to make the bounds an even multiple of the spacing. Use a value of 1 to enable maintaing the bounds, or 0 to use the default of keeping the spacing the same and growing the bounds slightly if needed.
- **EXPORT_BOUNDS_SNAP_PIXEL** - controls whether or not the top-left corner of an export bounds will be snapped to align with the closest pixel of the data being exported. Use MISC_OPT_VALUE="**1**" to enable this snapping, or MISC_OPT_VALUE="**0**" to disable (the default).
- **EXPORT_BOUNDS_SNAP_SPACING** - controls whether or not the top-left corner of an export bounds will be snapped to align with the closest sample spacing boundary. Use MISC_OPT_VALUE="**1**" to enable this snapping, or MISC_OPT_VALUE="**0**" to disable (the default).
- **TEMP_FOLDER** - specifies the base temp folder to use. Pass an empty string to reset to the default temp folder.
- **LOG_LEVEL** - specifies what level of messages are logged to the log file. Values range from **0** for only critical errors to **4** to log everything, including debug messages. The default is **1** which logs errors and critical errors.
- **LOG_FILENAME** - specifies the name of the file to log errors and warnings to (based on LOG_LEVEL).

- **MISC_OPT_VALUE** - specifies the value to set for the MISC_OPT option. For most options this will be **1** to enable or **0** to disable, but some of the options for the GM_MiscOpt type can take other *numbers* (i.e. unit settings) or even a *string value* (i.e. filenames for timing or log filename).

- **MISC_OPT_OLD_VALUE_VAR** - specifies the name of a variable to store the previous MISC_OPT value in. This is useful to later restore a changed value in a script. See example below.

- **DETAIL_OFFSET** - specifies the detail offset to use when determining which vector feature types to display based on zoom level. The valid range is *-150 to +150*. Values less than zero mean to draw feature types sooner than you normally would based on classification. Values larger than 0 mean to require zooming in further than normal to see a given feature type. This reflects the value of the detail slider on the Vector Display tab of the Configuration dialog. The default value of that slider is -150 which will always display all enabled vector types.

Specifying a Type/Lidar Filter/ shared Lidar Draw Mode

- **FILENAME** - full path to type filter GMF file to load

- **FILTER_TYPE** - specifies the type of features the filter file is for. The supported values area FILTER_TYPE=**AREA**, FILTER_TYPE=**LINE**, or FILTER_TYPE=**POINT**.

- **LIDAR_FILTER** - specifies a comma-separated list of Lidar class numbers to enable or disable. Provide a minus sign to remove the type from the filter rather than add it. The filter starts off with the current filter settings, but you can add ALL to enable everything or NONE to clear the filter, then add or remove stuff after that. For example, to specify a class filter with only types 2 and 3 enabled, use `LIDAR_FILTER="NONE,2,3"`. To get one with everything but classes 2 and 3, use `LIDAR_FILTER="ALL,-2,-3"`.

- **LIDAR_RETURN_FILTER** - specifies a comma-separated list of Lidar return types to enable or disable. Provide a minus sign (**-**)to remove the type from the filter rather than add it. The filter starts off with the current filter settings, but you can add **ALL** to enable everything or **NONE** to clear the filter, then add or remove stuff after that. For example, to specify a return filter with only unknown and first returns, use `LIDAR_RETURN_FILTER="NONE,0,1"`. To get one with everything but the first return, use `LIDAR_RETURN_FILTER="ALL,-1"`. The numeric values have the following meanings:
  - **0** - Unknown Returns
  - **1** - First Return
  - **2** - Second Return
  - **3** - Last Return
  - **4** - Single Return
  - **5** - First of Many Returns
  - **6** - Second of Many Returns
  - **7** - Third of Many Returns
  - **8** - Last of Many Returns

- **LIDAR_DRAW_MODE** - specifies the shared Lidar draw mode to use for any loaded layers that are set to use the shared global draw mode option if the Lidar module is active. The following values are supported:
  - **COLOR** - if the points have an associated RGB color, use that. Otherwise color by elevation.
  - **ELEV** - color by elevation of the point using the current elevation shader.
  - **INTENSITY** - color as a grayscale image by the intensity
  - **CLASS** - color by the point classification
  - **RETURN** - color by the return number

- **HEIGHT_ABOVE_GROUND** - color by the height above ground
- **POINT_SOURCE_ID** - color by the point source ID
- **POINT_INDEX** - color by the point index (i.e. order of points in file)
- **RETURN_HEIGHT_DELTA** - color by the difference in height between first and last point in multi-return samples
- **CIR** - color as Color Infrared (requires Lidar to have RGB + NIR)
- **NDVI** - color by calculated NDVI value (requires Lidar to have RGB + NIR)
- **NDWI** - color by calculated NDWI value (requires Lidar to have RGB + NIR)
- **DENSITY** - color by point density
- **WITHHELD** - color withheld points
- **KEY_POINT** - color model key points
- **OVERLAP** - color overlap points

SAMPLE

Example: Set an export option to snap to spacing, do an export, then restore the previous value

```
 // Enable the option to snap to a spacing interval and save old value
SET_OPT MISC_OPT="EXPORT_BOUNDS_SNAP_SPACING" MISC_OPT_VALUE="1" MISC_OPT_OLD_VALUE_VAR="PREV_
VAL_SPACING"
// Do the export
EXPORT_RASTER FILENAME="out.tif" TYPE=GEOTIFF
// Restore the old setting
SET_OPT MISC_OPT="EXPORT_BOUNDS_SNAP_SPACING" MISC_OPT_VALUE="%PREV_VAL_SPACING%"
```

LOAD_TYPE_FILTER

The LOAD_TYPE_FILTER command is deprecated as of Global Mapper v16.0.5. The SET_OPT command now handles both loading type filters and other generic global options. See that command for the list of parameters. The LOAD_TYPE_FILTER command will still work, but you should switch over to SET_OPT.

SET_VERT_DISP_OPTS

The SET_VERT_DISP_OPTS command allows you to modify the options used when rendering elevation layers, such as the shader to use, if any, as well as the lighting and water setup. The following parameters are supported by this command:

- **ENABLE_HILL_SHADING** - this setting controls whether or not hill shading (i.e. lighting, shadowing) will be done. Use **YES** to enable hill shading, and **NO** to disable hill shading.

- **SHADER_NAME** - this sets the name of the shader to use when rendering elevation data. This must be one of the names displayed in the shader drop down in Global Mapper, such as "**Atlas Shader**" or "**Global Shader**" or the name of a custom shader.

- **AMBIENT_LIGHT_LEVEL** - this sets the ambient lighting level. The valid range of values is [*0.0, 1.0*], with smaller numbers meaning completely black (i.e. no light) and 1.0 being full lighting.

- **VERT_EXAG** - this sets the vertical exaggeration to use when rendering elevation overlays. This effects the hill shading. The valid range of values is (*0.0, 100.0*].

- **LIGHT_ALTITUDE** - this sets the altitude angle of the light source that creates shadows on elevation data. The valid range of values is [*0.0, 90.0*], with 0.0 meaning a light source at the horizon and 90.0 meaning a light source directly overhead.

- **LIGHT_AZIMUTH** - this sets the direction angle of the light source that creates shadows on elevation data. The valid range of values is [*0.0, 360.0*), with 0.0 meaning a light source from the top of the screen (i.e. north), 90.0 meaning from the right (i.e. east), etc.

- **LIGHT_NUM_SOURCES** - this setting controls how many different directions of light the shading for terrain is computed from. By default you just get a light from the direction specified by LIGHT_AZIMUTH and LIGHT_ALTITUE, but if you specify a value for this parameter greater than 1 you get extra lights spacing around the circle. For example, using `LIGHT_NUM_SOURCES=4` gives you a light source at the LIGHT_AZIMUTH direction as well as 3 other sources 90, 180, and 270 degrees from that direction.

- **LIGHT_BLENDING_ALGORITHM** - this setting controls how the shading for terrain is computed by blending light sources from multiple directions (as specified by LIGHT_NUM_SOURCES). The following values are supported:

- **0** - Average shadow from each light
- **1** - Maximum shadow from any light
- **2** - Minimum shadow from any light
- **3** - Weighted average shadow from each light, with specified light azimuth getting highest weight

- **SHADE_DARKNESS** - this sets the minimum black level that a shadow can create. The valid range of values is [*0,255*], with 0 allowing complete blackness from a shadow, and 255 allowing no shadow at all.

- **SHADE_HIGHLIGHT** - this sets the level of white highlight applied to terrain areas directly facing the sun-/light angle. The valid range of values is [*0,255*], with 0 applying no highlight and 255 making the direct areas always brightened completely to white.

- **ENABLE_WATER** - this setting controls whether or not water will be displayed on top of elevation values at or below the currently configured water level. Use **YES** to enable water display, and **NO** to disable water display.

- **WATER_COLOR** - this setting controls the color that water drawn on top of elevation data is rendered in. The format of this value is *RGB(<red>,<green>,<blue>)*. For example, to use a water color of blue, use `WATER_COLOR=RGB(0,0,255)`.

- **WATER_LEVEL** - this setting specifies the height (in meters) below which water should be displayed if enabled.

- **WATER_ALPHA** - this setting controls how "see through" the water is when displayed. The valid range of values is [*0,255*], with 0 meaning the water is completely "see through", i.e. invisible, and 255 meaning that the water color is completely opaque such that you can't see any of the shaded relief below it.

- **SLOPE_ALGORITHM** - this setting controls how the slope at a given location in a layer is calculated. The following values are supported:
  - **0** - (**Default**) Average Maximum of Slope to 4 Non-Diagonal Adjacent Samples - This method computes the slope from the cell center to the left and right, and top and bottom samples, then combines those to get a single slope value.
  - **1** - Average Maximum of Slope to All 8 Adjacent Samples - This method computes the slope from the cell center to the left and right, and top and bottom samples, then combines those to get a single slope value.
  - **2** - Maximum Slope to 4 Non-Diagonal Adjacent Samples - This method computes the slope from the cell center to the left, right, top, and bottom samples, then uses the maximum of those 4 slope values.
  - **3** - Maximum Slope to All 8 Adjacent Samples - This method computes the slope from the cell center to each of the 8 adjacent cell centers, then uses the maximum of those 4 slope values.

- **DAYLIGHT_SHADER_COLOR** - this setting controls the color that is used when rendering terrain using the Daylight Shader. The format of this value is *RGB(<red>,<green>,<blue>)*. For example, to use a color of red, use `WATER_COLOR=RGB(255,0,0)`.

SET_VIEW

The SET_VIEW command sets the current view bounds. Use this in workspace files that run in the context of the main map view.

The following parameters are supported by the command:

- **GLOBAL_BOUNDS** - specifies the combine bounds in units of the current global projection. There should be 4 values in a comma-delimited list following the parameter name. The values should be in order of *minimum x, minimum y, maximum x, maximum y*.

- **GLOBAL_BOUNDS_SIZE** - specifies the combine bounds in units of the current global projection. There should be 4 values in a comma-delimited list following the parameter name. The values should be in order of *minimum x, minimum y, width in x, width in y*.

- **LAT_LON_BOUNDS** - specifies the combine bounds in latitude/longitude degrees. There should be 4 values in a comma-delimited list following the parameter name. The values should be in order of *west-most longitude, southern-most latitude, eastern-most longitude, northern-most latitude*.

- **LAYER_BOUNDS** - specifies that the operation should use the bounds of the loaded layer(s) with the given filename. For example, to export to the bounds of the file "c:\test.tif", you would use `LAYER_BOUNDS-S="c:\test.tif"`. Keep in mind that the file must be currently loaded.

- **LAYER_BOUNDS_EXPAND** - specifies that the operation should expand the used LAYER_BOUNDS bounding box by some amount. The amount to expand the bounding rectangle by should be specified in the current global projection. For example, if you have a UTM/meters projection active and want to expand the bounds retrieved from the LAYER_BOUNDS parameter by 100 meters on the left and right, and 50 meters

on the top and bottom, you could use `LAYER_BOUNDS_EXPAND="100.0,50.0"`. You can also specify a single value to apply to all 4 sides, or supply 4 separate values in the order *left,top,right,bottom*.

- **SNAP_BOUNDS_TO_MULTIPLE** - specifies that the top-left corner of the bounding box for the operation should be snapped to a multiple of the given value. For example, using `SNAP_BOUNDS_TO_MULTIPLE=1` will snap the top-left corner to the nearest whole number. The values will always go smaller for X/east-ing/longitude and larger to Y/northing/latitude so you always get at least what is requested.

- **SNAP_BOUNDS_TO_SPACING** - specifies that the top-left corner of the bounding box for the operation should be snapped to a multiple of the resolution of the operation. For example, if you are exporting at 5 meter spacing, the top left corner will be snapped to the nearest multiple of 5. Use SNAP_BOUNDS_TO_ SPACING=**YES** to enable or SNAP_BOUNDS_TO_SPACING=**NO** to disable. If not provided, the global set-ting for snapping exports to the nearest sample spacing boundary from the Advanced section of the Gen-eral tab of the Configuration dialog will be used.

- **USE_EXACT_BOUNDS** - specifies that the exact bounds that were defined in the command should be used. Generally, when the bounds specified in a command are not the same as the data bounds, the com-mand uses the intersection between the two. When USE_EXACT_BOUNDS=**YES** is specified, the command will use the bounds as specified, instead of the intersection.

Specify Bounding Box for Operation

SAVE_CURRENT_VIEW

The SAVE_CURRENT_VIEW command saves the current view window for later restoration using the RESTORE_ LAST_SAVED_VIEW command. This command does not take any parameters.

RESTORE_LAST_SAVED_VIEW

The RESTORE_LAST_SAVED_VIEW command restores the last view saved with the SAVE_CURRENT_VIEW com-mand (or the last view saved with the View->Save Current View menu command in the Global Mapper user inter-face). This command does not take any parameters.

SET_BG_COLOR

The SET_BG_COLOR command sets the color to use for any background pixels when rendering layers. The fol-lowing parameters are supported by the command.

- **COLOR** - specifies the background color to use. The color should be specified as *RGB(<red>,<-green>,<blue>)*. For example, to make white the background color, use `COLOR=RGB(255,255,255)`.

SHOW_3D_VIEW

The SHOW_3D_VIEW command displays the 3D view window with the loaded data.

The following parameters are supported by the command:

- **MAXIMIZE** - specifies whether or not the 3D view window should be maximized or not. Use MAXIMIZE=**YES** to force the window to be maximized when displayed.

VIEW_LAYOUT

The VIEW_LAYOUT command allows you to define the multi-view map layout defining where various map views are placed in the user interface.

The VIEW_LAYOUT command consists of a single command line followed by a series of lines with the contents of a .gm_views file, like you would save from the Map View Manager dialog. You can also provide a .gm_views in the FILENAME parameter to just reference an external file rather than embedding the map view layout definition dir-ectly in the script file.

The VIEW_LAYOUT command is terminated with a single line containing only the text **END_VIEW_LAYOUT**.

For a sample of the VIEW_LAYOUT command in use, set up the view layout you want using the Map View Manager, then save a Global Mapper workspace file from the File->Save Workspace menu command. Open the resulting .gmw file in an editor and you can see how the VIEW_LAYOUT command is used at the bottom of the file to define

the map layout so long as you haven't disabled the save/restore of the multi-view layout to workspace files in the Advanced section of the General tab of the Configuration dialog.

The following parameters are required by the VIEW_LAYOUT command.

- **FILENAME** - specifies the name of a .gm_views file to use rather than embedding it.

MAP_LAYOUT

The MAP_LAYOUT command allows you to define the map layout settings to use for the map display, including margins, scale bar, legend, etc.

The MAP_LAYOUT command consists of a single command line followed by a series of lines with the contents of a .gm_layout file, like you would save from the Map Layout dialog. You can also provide a .gm_layout in the FILENAME parameter to just reference an external file rather than embedding the map layout directly in the script file.

The MAP_LAYOUT command is terminated with a single line containing only the text **END_MAP_LAYOUT**.

For a sample of the MAP_LAYOUT command in use, load some data, then save a Global Mapper workspace file from the File->Save Workspace menu command. Open the resulting .gmw file in an editor and you can see how the MAP_LAYOUT command is used at the bottom of the file to define the map layout.

The following parameters are required by the MAP_LAYOUT command.

- **FILENAME** - specifies the name of a .gm_layout file to use rather than embedding it.

Import/ Open Data

IMPORT

The IMPORT command imports a data file for later use. The following parameters are supported by the command.

- **FILENAME** - full path to file to load the data from. This can also be the URL (http: or ftp:) for a file on a web site that you want Global Mapper to download and load. You can include wildcards ('*' or '?') in the FILENAME (v16.0.5. or later) to load all files matching a particular mask. In v17.0.4 or later you can provide workspace or script files (.gmw and .gms) via an IMPORT command rather than using EMBED_SCRIPT.

- **TYPE**

  The following import formats may be specified:

  - **AUTO** - automatically determine the type (default).
  - **ACE** - Altimetry Corrected Elevation (ACE) format file.
  - **ARCASCIIGRID** - Arc ASCII Grid format file.
  - **ARCBINARYGRID** - Arc Binary Grid format file.
  - **AVC** - Arc Vector Coverage format file.
  - **BIL** - BIL format file.
  - **BSB** - BSB format file (usually has .KAP extension).
  - **BT** - a BT (Binary Terrain) format grid file.
  - **CANADA3D** - Canada 3D format file.
  - **COMPEGPS** - a CompeGPS RTE, TRK, or WPT file.
  - **CPS3** - a CPS-3 grid file.
  - **CTM_DEM** - a CTM DEM format file.
  - **DBF** - DBase file with point features.
  - **DHM25** - a Swiss DHM terrain format file.
  - **DLGO** - USGS DLG-O
  - **DGN** - MicroStation DGN files earlier than v8.
  - **DIVAGIS_GRID** - a DIVA GIS grid format file.
  - **DMDF** - a Digital Map Data Format (DMDF) format file.
  - **DOQQ** - USGS DOQ in JPEG format.
  - **DTED** - Digital Terrain Elevation Data (DTED) format.
  - **DXF** - DXF format.
  - **ECW** - ER Mapper Compressed Wavelet (ECW) format file.
  - **EMF** - a Windows Enhanced Metafile (EMF) format file.
  - **ENVI DEM** - ENVI DEM format file.
  - **ERDAS** - Erdas Imagine format file.
  - **ERDAS_GIS** - Erdas GIS format file.
  - **ERM_GRID** - ERM grid format file.
  - **ESRI_PGEO** - ESRI personal geodatabase format file.
  - **ETOPO2** - ETOPO2 format file.
  - **FAST_L7A** - a Landsat FAST L7A format file.
  - **FILE_GDB** - an ESRI File Geodatabase
  - **FLOATGRID** - FLOAT/GRID format file.
  - **GEOSOFT_GRID** - a Geosoft Binary Grid format file.

- **GEOTIFF** - GeoTIFF format file.
- **GIF** - a GIF format file with associated world file.
- **GLOBAL_MAPPER_CATALOG** - a Global Mapper Map Catalog file.
- **GLOBAL_MAPPER_GRID** - a Global Mapper Grid format file.
- **GML** - a GML format file.
- **GNIS** - Geographics Names Information Service (GNIS) file
- **GPS_TRACKMAKER** - a GPS TrackMaker format file.
- **GXF** - Geosoft Grid ASCII (GXF) file
- **GPX** - GPS eXchange Format file
- **HDF** - an HDF format raster or grid file, like ASTER DEM or ASTER VNIR imagery.
- **HELAVA_DEM** - a Helava DEM file
- **HYDRA_GRID** - a Hydra Grid file
- **HYPACK_MATRIX** - a Hypack Matrix format file.
- **IBCAO** - Arctic bathymetry in NetCDF format.
- **IDRISI_RASTER** - Idrisi raster/elevation format file.
- **IGF_DIS** - an IGF-DIS format file
- **INM_3TX** - an INM 3TX grid file.
- **INTERGRAPH_COT** - Intergraph COT format file.
- **JDEM** - a Japanese DEM .mem file.
- **JPEG** - a JPEG file with an associated world file.
- **JPEG2000** - a JPEG 2000 file
- **JPGIS** - JPGIS (Japanese DEM) XML file.
- **KML** - a KML/KMZ file.
- **LIDAR_LAS** - a LAS file with LIDAR data.
- **LOWRANCE_USR** - a Lowrance USR format file.
- **LULC** - USGS Land Use and Land Cover vector data file.
- **MAPINFO** - a MapInfo MIF/MID of TAB/MAP vector data collection.
- **MAPMAKERTERRAIN** - a MapMaker terrain file
- **MARPLOT_MIE** - a MarPlot MIE format file.
- **MAPTECH** - a MapTech BSB, Topo, or Aerial format file.
- **MICRODEM_DEM** - a MicroDEM-created DEM file
- **MICROPATH_3CD** - a Micropath 3CD grid file.
- **MICROPATH_DEM** - a Micropath DEM file
- **MRSID** - a LizardTech MrSID image file.
- **NetCDF** - a NetCDF format file.
- **NIMA_GNS** - a NIMA GNS format file.
- **NITF** - NITF format imagery
- **NOS_GEO** - a NOS/GEO format chart file.
- **NTF_GRID** - a NTF grid format file.
- **OPTIMI_GRID** - an Optimi terrain or clutter grid format file.
- **OZI** - an OziExplorer format waypoint (WPT) or track (PLT) file.
- **PCX** - a PC Paintbrush PCX format file.
- **PCX5** - a Garmin PCX5 format waypoint (WPT) or track (TRK) file.
- **ROCKWORKS_GRID** - a RockWorks Grid format file.
- **RPF** - Raster Product Format database, like CADRG.
- **RPF_FRAME** - single frame from a Raster Product Format database, like CADRG.
- **S57** - a S-57 chart file
- **SDTS** - a SDTS transfer
- **SEGP1** - a SEGP1 seismic shotpoint file.
- **SHAPEFILE** - an ESRI Shapefile.
- **SURFERGRID** - a Surfer grid format file.
- **SWEDISHDEMGRID** - a Swedish DEM grid format file.
- **TERRASCAN** - a TerraScan LIDAR format file.
- **TIGER_LINE** - a Tiger/Line format file.
- **TRMM_GRID** - a TRMM precipitation grid file.
- **USGS_DEM** - a native format USGS DEM file.
- **VPF** - a Vector Product Format file such as VMAP or DNC data
- **VULCAN_3D** - a Vulcan3D triangulation file
- **WASP_RESOURCE_GRID** - a WaSP resource grid file
- **ZMAP_PLUS** - a Geographix Zmap Plus+ format file.
- **PROMPT_IF_TYPE_UNKNOWN** - set to **NO** if you don't want the user to be prompted to select a file type if the type cannot automatically be determined (useful when looping).

Shared Import Parameters

These parameters are shared across import and layer options commands to set layer properties.

- **HIDDEN** - set to **YES** to cause this overlay to be hidden from view after it is loaded. The default is to show the overlay.

- **LAYER_DESC** - specifies a description to use for the layer when displaying it in the Overlay Control Center. This overrides the default description based on the filename or other information within the file.

- **LAYER_GROUP** - specifies the name of the group for the layer in the Control Center. To include multiple layers of grouping put the string <sub> in between levels. For example to make a group with 2 levels of nesting, use LAYER_GROUP="Top Level<sub>Next Level".

- **ALLOW_SELECTION** - set to **NO** to disable selection of features from this layer using either the Feature Info or Digitizer Tools.

- **ALLOW_EXPORT** - set to **NO** to disable export from this layer.

- **LOAD_FLAGS** - contains flags for any import options that you were prompted for when loading the file, such as if you have a .tif file that you were prompted to select as elevation or raster. Also things like the coverages and tile sets for VPF layers. To see how to set these if you are writing a script, load a file with the settings that you want in the main user interface and then save a workspace, then examine the IMPORT command in the .gmw file for that file and see how the LOAD_FLAGS were set.

- **METADATA_FILENAME** - specifies full path and filename of a file to display the contents of on the Metadata dialog for a layer. The file can be any simple displayable text format, including text and XML.

- **METADATA_URL** - specifies a URL to a displayable web file (including HTML web page or XML document) to show on the Metadata dialog for a layer.

- **CODE_PAGE**- specifies the code page to use when interpreting text from this layer. By default if the file doesn't specify a code page the current system code page will be used. Use the code page number, or the text UTF-8 (number 65001).

- **ALT_MODE** (vector only) - altitude mode specifies how the 3D viewer should interpret z-values in the vector features of an layer, relative to terrain. Altitude mode may also be set in an individual feature, in which case it overrides the layer setting. The following values are supported:

    - **UNSPECIFIED** - Altitude mode is determined by either the setting in the feature, or if unspecified, the setting in the 3D viewer
    - **ABSOLUTE** - treat z-values as absolute elevations, ignoring any terrain
    - **RELATIVE_TO_GROUND** - treat z-values as distances above the terrain
    - **RELATIVE_TO_SEA_FLOOR** - treat z-values as distances above the sea floor (currently implemented as RELATIVE_TO_GROUND)
    - **CLAMP_TO_GROUND** - ignore z-values, and clamp the feature to the terrain
    - **CLAMP_TO_SEA_FLOOR** - ignore z-values, and clamp the feature to the sea floor (currently implemented as CLAMP_TO_GROUND)
    - **DEPTH** - treat z-values as absolute depths, ignoring any terrain

- **ZOOM_DISPLAY** - specifies when the map should be displayed and when it should be hidden based on the display zoom scale. This command will be formatted as a name from the list, below followed by 2 numeric paramters. For example, use ZOOM_DISPLAY="SCALE,25000,0" to have a map display only when zoomed in below 1:25000 scale.

    - **ALWAYS** - always display the map. The numeric parameters are ignored.
    - **PERCENT** - display the map when the map bounding box is a certain percentage of the screen size. For example, use ZOOM_DISPLAY="PERCENT,0.10,0" to display the map when its bounding box is at least 10% of the screen size.
    - **PIXEL_SIZE** - display the map when each display pixel is less than some number of meters in size. For example, use PIXEL_SIZE="SCALE,10,0" to display the map when the current display resolution is 10 meters per pixel (or less/higher resolution).
    - **SCALE** - display the map when the current display is at or below a certain scale. For example, use ZOOM_DISPLAY="SCALE,25000,0" to display the map when the current draw scale is at or below 1:25000.
    - **SCALE_RANGE** - display the map when the current display is below a range of scale value. For example, use ZOOM_DISPLAY="SCALE_RANGE,25000,100000" to display the map when the current draw scale is between 1:25000 and 1:100000.

- **PROJ** - special Projection Specification type of parameter that specifies the projection to use for the file. This will override any projection information stored in the file.

- *PROJ_NAME* (DEPRECATED use PROJ instead) - specifies the name of the projection to use for this file (this will override any projection information stored in the file). This name must have been defined with a prior DEFINE_PROJ command.

- *PROJ_FILENAME* (DEPRECATED use PROJ instead)- specifies the name of the projection (.prj) file to use for this file (this will override any projection information stored in the file).

- ***PROJ_EPSG_CODE*** (DEPRECATED use PROJ instead) - specifies the numeric EPSG projection code that defines the projection for this file (this will override any projection information stored in the file). For example, use PROJ_EPSG_CODE=26715 to define a UTM zone 15 projection with NAD27 as the datum and meters as the units.

- **PROMPT_IF_PROJ_UNKNOWN** - set to **NO** if you don't want the user to be prompted to select a projection if the projection of the file cannot be automatically determined.

- **USE_DEFAULT_PROJ** - specifies that if no projection can be automatically determined for a layer that the default projection selection should be used rather than prompting the user. Use USE_DEFAULT_PROJ=**YES** to enable. The default projection uses the first valid option from the following, including a check for linear versus angular numeric ranges:
    - Projection of any files loaded from the same folder
    - Last projection user selected on a projection dialog in this session
    - Current view projection
    - Projection from default.prj in global_mapper.exe path
    - Projection from default.prj in User Settings File path
    - Last projection user selected on a projection dialog in previous session of GM
    - Default UTM/15N/NAD83 projection

- **USE_DEFAULT_POS** - specifies that if no position data for a raster layer can be automatically determined that a default position should be chosen so that it displays. Use USE_DEFAULT_POS=**YES** to enable.

- **PICTURE_POS** - specifies that the image should be loaded as a 'picture point' that displays the image when you select the point with the Feature Info Tool. The value should contain the *X and Y coordinates* (in the projection specified for the layer). For example to place the value at 30N 95W with the projection set as PROJ_EPSG_CODE=4326 you can use `PICTURE_POS="-95.0,30.0"`.

- **LOAD_HIDDEN_PDF_LAYERS** - for PDF import, specifies that if no layer prompt is provided that hidden layers should be loaded automatically. Use LOAD_HIDDEN_PDF_LAYERS=**YES** to enable.

Elevation Parameters

Parameters for display and interpretation of elevation values in terrain layers. See also Raster Parameters below for additional shared parameters.

- **ELEV_FIELD** - specifies the name of the attribute field to use as the elevation value for the vector features in a file

- **ELEV_UNITS** - specify elevation units to use for this file if it contains gridded elevation data and also for vector feature elevations that don't have a unit embedded in the elevation value. Valid values are as follows:
    - **FEET** - elevations in US feet
    - **DECIFEET** - elevations in 10ths of US feet
    - **METERS** - elevations in meters
    - **DECIMETERS** - elevations in 10ths of meters
    - **CENTIMETERS** - elevations in centimeters

- **ELEV_OFFSET** (elevation only) - specifies the offset in meters to apply to each elevation value in the layer. This allows you to vertically shift a layer to match other layers.

- **ELEV_POWER** (elevation only) - specifies the power value to apply to each elevation value in the layer. For example a value of **2.0** would square each elevation value before applying a scale and adding the offset. Default to **1.0** (no power).

- **ELEV_SCALE** (elevation only) - specifies the scale value to apply to each elevation value in the layer. This allows you to vertically scale a layer to match other layers. Default to **1.0** (no scaling).

- **MIN_ELEV** (elevation only) - specifies the minimum elevation (meters) to treat as valid when rendering this layer. Any elevations below this value will be treated as invalid and not be drawn or exported.

- **MAX_ELEV** (elevation only) - specifies the maximum elevation (meters) to treat as valid when rendering this layer. Any elevations above this value will be treated as invalid and not be drawn or exported.

- **CLAMP_ELEVS** (elevation only) - if a MIN_ELEV and /or MAX_ELEV value is specified, setting this to **YES** will make any valid elevation values outside of the specified range be clamped to the new range value rather than treated as invalid.

- **VOID_ELEV** (elevation only) - specifies the elevation (meters) to replace any void areas in the layer with. If not specified, the void areas will be transparent.

- **SHADER_NAME** (elevation only) - this sets the name of the shader to use when rendering the gridded elevation data for this layer. Use this to override use of the shared default shader just for this layer. This

must be one of the names displayed in the shader drop down in Global Mapper, such as **"Atlas Shader"** or **"Global Shader"** or the name of a custom shader.

Raster Parameters

Parameters for display of imagery. Some of the below parameters are also supported for elevation layers.

- **SAMPLING_METHOD** (elevation and raster only) - specifies the sampling method to use when res-ampling this layer.

  The following values are supported
  - **NEAREST_NEIGHBOR** - use the nearest neighbor sampling method
  - **BILINEAR** - use bilinear interpolation
  - **BICUBIC** - use bicubic interpolation
  - **BOX_2X2** - use a 2x2 box average
  - **BOX_3X3** - use a 3x3 box average
  - **BOX_4X4** - use a 4x4 box average
  - **BOX_5X5** - use a 5x5 box average
  - **BOX_6X6** - use a 6x6 box average
  - **BOX_7X7** - use a 7x7 box average
  - **BOX_8X8** - use a 8x8 box average
  - **BOX_9X9** - use a 9x9 box average
  - **MAX_2X2** - (terrain layers only) use maximum value found in 2x2 box
  - **MAX_3X3** - (terrain layers only) use maximum value found in 3x3 box
  - **MAX_4X4** - (terrain layers only) use maximum value found in 4x4 box
  - **MAX_5X5** - (terrain layers only) use maximum value found in 5x5 box
  - **MAX_6X6** - (terrain layers only) use maximum value found in 6x6 box
  - **MAX_7X7** - (terrain layers only) use maximum value found in 7x7 box
  - **MAX_8X8** - (terrain layers only) use maximum value found in 8x8 box
  - **MED_2X2** - (terrain layers only) use median value found in 2x2 box
  - **MED_3X3** - (terrain layers only) use median value found in 3x3 box
  - **MAX_9X9** - (terrain layers only) use maximum value found in 9x9 box
  - **MED_4X4** - (terrain layers only) use median value found in 4x4 box
  - **MED_5X5** - (terrain layers only) use median value found in 5x5 box
  - **MED_6X6** - (terrain layers only) use median value found in 6x6 box
  - **MED_7X7** - (terrain layers only) use median value found in 7x7 box
  - **MED_8X8** - (terrain layers only) use median value found in 8x8 box
  - **MED_9X9** - (terrain layers only) use median value found in 9x9 box
  - **MIN_2X2** - (terrain layers only) use minimum value found in 2x2 box
  - **MIN_3X3** - (terrain layers only) use minimum value found in 3x3 box
  - **MIN_4X4** - (terrain layers only) use minimum value found in 4x4 box
  - **MIN_5X5** - (terrain layers only) use minimum value found in 5x5 box
  - **MIN_6X6** - (terrain layers only) use minimum value found in 6x6 box
  - **MIN_7X7** - (terrain layers only) use minimum value found in 7x7 box
  - **MIN_8X8** - (terrain layers only) use minimum value found in 8x8 box
  - **MIN_9X9** - (terrain layers only) use minimum value found in 9x9 box

- *ANTI_ALIAS* [DEPRECATED - use SAMPLING_METHOD instead] (elevation and raster only) - specifies whether to remove jagged edges by making a subtle transition between pixels. Turning off this option helps maintain the hard edges of the pixels as they are rasterized. Use ANTI_ALIAS=YES to turn on. Anything else turns it off.

- **TRANSLUCENCY** (elevation and raster only) - specifies the level of translucency (i.e. how "see-through" the layer is). Value values range from *0 to 512*, with 0 meaning the layer is completely transparent (i.e. invisible) and 512 meaning the layer is completely opaque (this is the default).

- **IGNORE_ALPHA** (raster only) - specifies that an embedded alpha channel in an image should be ignored. This is useful for images that have incorrect alpha channels. Use IGNORE_ALPHA=**YES** to enable.

- **BLEND_MODE** (elevation and raster only)- specify blend mode to use for combining this overlay and any previously loaded overlays
  - **NO_BLEND** - no blending is done, this is the default
  - **MULTIPLY**
  - **SCREEN**
  - **OVERLAY**
  - **HARD_LIGHT**
  - **COLOR_BURN**
  - **COLOR_DODGE**
  - **DARKEN**
  - **LIGHTEN**
  - **DIFFERENCE**
  - **EXCLUSION**

- **APPLY_COLOR**
- **APPLY_COLOR_REVERSE**
- **KEEP_RED**
- **KEEP_GREEN**
- **KEEP_BLUE**
- **SPOT_NATURAL_COLOR_SPOT_TO_NATURAL**
- **PSEUDO_NATURAL_COLOR_CIR_TO_NATURAL**
- **COLOR_TO_GRAYSCALE**

- **FEATHER_BLEND_EDGES** (raster only) - specifies that the layer should be feature-blended around one or more ledges. This is a numeric bitfield value. Add (sum) the following values to enable blending on that edge:

  - **1** - blend top edge
  - **2** - blend bottom edge
  - **4** - blend left edge
  - **8** - blend right edge
  - **32** - just crop to feather boundary rather than feathering
  - **64** - feather outside polygon edge

  For example, to blend all edges, use FEATHER_BLEND_EDGES=**15**. The FEATHER_BLEND_SIZE parameter is used to specify how many pixels to blend on each blended edge.

- **FEATHER_BLEND_SIZE** (raster only) - specifies the size in pixels to use for a blend boundary.

- **FEATHER_BLEND_POLY** (raster only) - specifies the name of a previously defined shape from "DEFINE_ SHAPE" on page 20 to feather too. You can also use FEATHER_BLEND_POLY=**COVERAGE** to calculate the polygonal coverage of the layer and feather blend to that automatically.

- **FEATHER_BLEND_POLY_FILE** (raster only) - specifies that the polygon boundary to feather blend this layer against should come from the specified file.

- **BAND_SETUP** (raster only) - specifies what bands of data from the raster file being loaded should be used to populate the red, green, and blue color channels when displaying the image. This is useful for multi-spectral imagery which may have more than 3 color bands. The default band setup will be to use the first three bands as follows: BAND_SETUP="**0,1,2**". Note that not all raster formats support specifying a non-default band setup.

- **CONTRAST_MODE** (raster only) - specifies the type of contrast adjustment to apply to the data.

  - **NONE** - no contrast adjustment applied (this is the default)
  - **PERCENTAGE** - apply a percentage contrast adjustment. The CONTRAST_STRETCH_SIZE parameter can be used to override the number of standard deviations from the mean to stretch to.
  - **MIN_MAX** - apply a min/max contrast stretch, stretching the available range of values in each color band to the full range of 0-255. For imagery which contains both black and white, this will have no affect.

- **CONTRAST_SHARED** (raster only) - specifies whether or not the contrast adjustment for this layer will share the adjustment with other contrast-adjusted layers in order to ensure a consistent modification across layers. Use CONTRAST_SHARED=**YES** to enable contrast sharing.

- **CONTRAST_STRETCH_SIZE** (raster only) - specifies the number of standard deviations from the mean to use in a PERCENTAGE contrast adjustment. The default is 2.0.

- *AUTO_CONTRAST* (raster only) - DEPRECATED, use CONTRAST_MODE instead - specifies whether to automatically calculate and apply a 2 standard deviation contrast adjustment to the image. Use AUTO_ CONTRAST=YES to turn on. Anything else turns it off.

- *COLOR_INTENSITY*(DEPRECATED use COLOR_INTENSITY_FULL parameter) (elevation and raster only). - specifies the color intensity to use when adjusting the brightness of pixels in the overlay. Valid values range from *0 to 20*, with 0 being completely black, 10 being no alteration, and 20 being completely white. For example, to make an image slightly darker, you could use COLOR_INTENSITY=7.

- **COLOR_INTENSITY_FULL** (elevation and raster only) - specifies the color intensity to use when adjusting the brightness of pixels in the overlay. Valid values range from *0 to 512*, with 0 being completely white, 256 being no alteration, and 512 being completely black. For example, to make an image slightly darker, you could use COLOR_INTENSITY=300. (NOTE: This parameter replaces the COLOR_INTENSITY parameter).

- **TEXTURE_MAP** (raster only) - specifies that this image should be draped over any elevation data loaded before it. Use TEXTURE_MAP=**YES** to turn on. Anything else turns it off.

- **PALETTE_NAME** (palette-based raster only) - specifies the filename of a recognized palette file to override the default colors in this layers palette or a palette previously defined with the DEFINE_PALETTE command. Use this to change the color interpretation of palette indices.

- **TRANSPARENT_COLOR** (elevation and raster only) - specifies the color to make transparent when rendering this overlay. The color should be specified as *RGB(<red>,<green>,<blue>)*. For example, to make white the transparent color, use `TRANSPARENT_COLOR=RGB(255,255,255)`. If you do not wish any color to be transparent, do not use this parameter. Optionally, if the image that you are making transparent uses a palette for the colors, you can specify a palette index in the following format: *INDEX(<0-based palette index>)*. For example, to make the second color in the palette transparent, use `TRANSPARENT_COLOR=INDEX(1)`.

- **TRANSPARENT_COLOR_DIST** - for layers that have specified a color to make transparent, this parameter allows you to specify how far a color in the layer has to be from the specified TRANSPARENT_COLOR value to be treated as transparent as well. The default value of **0** means that the colors have to exactly match for the pixel to be treated as transparent. Larger values (*up to* **256**) allow larger distances between the layer color and the transparent color. This is useful for lossy formats, like JPEG.

- **COLOR_GRADE** (raster only) - specifies the color grading values to use for this layer (as configured on the Color Grade options dialog tab). This should be a comma-delimited list with the saturation value (from **0**-**1**) first, then the input and output range for the red, green, and blue color channels, as follows: COLOR_GRADE=*saturation,red_in_start,red_in_end,red_out_start,red_out_end,...,blue_out_end*

- **CLIP_COLLAR** (raster only) - specifies whether to clip the collar off of the image. The following values are supported for cropping:
  - **NONE** - no collar cropping is performed.
  - **AUTO** - automatically remove a USGS DRG-style collar or a 3.75 DOQQ collar
  - **LAT_LON** - crop the collar to a a specified set of bounds specified in arc degrees in the native datum of the layer. The bounds should be specified using the CLIP_COLLAR_BOUNDS parameter.
  - **NATIVE** - crop the collar to a specified set of bounds specified in the native projection system and datum of the layer. The bounds should be specified using the CLIP_COLLAR_BOUNDS parameter.
  - **PIXELS** - crop a given number of pixels off of each side of the layer. The number of pixels to remove from each side should be specified using the CLIP_COLLAR_BOUNDS parameter.
  - **SNAP_DEGREES** - crop the collar by snapping each edge to a specified degree boundary specified in arc degrees in the native datum of the layer. The bounds should be specified using the CLIP_COLLAR_BOUNDS parameter. For example to crop the west and east edges to a half degree boundary and the north and south edges to a one degree boundary, use the following: `CLIP_COLLAR_BOUNDS=0.5,1.0,0.5,1.0`.
  - **POLY** - crop to a polygon provided with the CLIP_COLLAR_POLY parameter.

- **CLIP_COLLAR_BOUNDS** (raster only) - specifies the bounds of the collar to be clipped off when the CLIP_COLLAR parameter is enabled. The coordinates should be specified in arc degrees, native layer coordinates, or pixel counts as a comma-delimited list going *west,south,east,north*. For example, `CLIP_COLLAR_BOUNDS=34.25,-109.0,34.375,-108.875`.

- **CLIP_COLLAR_POLY** (raster only) - specifies the name of the previously defined shape (with the [DEFINE_SHAPE](#) command) to crop the layer to when the CLIP_COLLAR=POLY parameter is used. The coordinates in the shape must have been defined in the native projection system of the layer being loaded. Unless you provide CLIP_COLLAR_POLY_SIMPLIFY=NO, the clip polygon will be simplified to 1/10th of a pixel resolution to reduce the size of the crop polygon for faster cropping without noticeably changing the shape.

- **CLIP_COLLAR_POLY_EXCLUDE** (raster only) - specifies that the crop to the polygon specified with CLIP_COLLAR_POLY should keep all parts of the layer outside the crop polygon(s) rather than what is inside the polygon(s). Add CLIP_COLLAR_POLY_EXCLUDE=**YES** to enable this behavior.

- **CLIP_COLLAR_POLY_SIMPLIFY** - specifies that the clip polygon will be simplified to 1/10th of a pixel resolution to reduce the size of the crop polygon for faster cropping without noticeably changing the shape. Enabled by default, use CLIP_COLLAR_POLY_SIMPLIFY=**NO** to disable.

- **CLIP_COLLAR_POLY_PIXEL** (raster only) - specifies that the coordinates in the crop polygon from the CLIP_COLLAR_POLY parameter are in pixel relative coordinates for the layer rather than in the native system of the layer. Use this if you need to crop a layer to a particular boundary in known pixel coordinates. The coordinates will convert to native layer coordinates on load.

Vector Parameters

- **VIDEO_FILENAME** - specifies the full path and filename or URL for a video file to associate with the layer.

This video can then be displayed for selected point or line features with a video timestamp.

Vector Label Parameters

The parameters below allow specifying how to create display labels for vector layers.

- **LABEL_FIELD** - specifies the *name of the attribute field* to use as the label attribute for the features in the file. By default the attribute-based labeling will only be applied to those features that don't already have a label, but if the LABEL_FIELD_FORCE_OVERWRITE attribute is set to YES then all features will have their labels replaced. If you want to build the label from multiple attributes, separate them with '>+<' in the file, like `LABEL_FIELD='RD_PREFIX>+<RD_NAME>+<RD_SUFFIX'`.

- **LABEL_CUSTOM_DEF** - specifies a custom free-form string describing how to form the display labels for this layer. This can include embedded attribute values as *%ATTR_NAME%*.

- **LABEL_FIELD_FORCE_OVERWRITE** - specifies that the LABEL_FIELD or LABEL_CUSTOM_DEF attribute value should be applied to all feature labels, not just those that don't already have labels. Use LABEL_ FIELD_FORCE_OVERWRITE=**YES** to enable.

- **SHOW_LABELS** - specifies whether or not labels are shown for features in this layer, assuming they would be otherwise shown. The default is SHOW_LABELS=**YES**. Use SHOW_LABELS=**NO** to disable the display of labels for this layer regardless of other settings.

- **LABEL_PREFIX** - specifies the prefix to prepend to attribute-based labels

- **LABEL_SUFFIX** - specifies the suffix to append to attribute-based labels

- **LABEL_PRECISION** - value is an integer indicating the number of decimal digits to use. This applies to numeric labels.

- **LABEL_REMOVE_TRAILING_ZEROS** - This removes the trailing zeros to the right of the decimal place in numeric labels. This can be specified by listing the parameter alone, or accepts boolean values.

- **LABEL_USE_SCIENTIFIC_NOTATION** - Display the number in scientific notation. This accepts boolean values, or can be called by listing the parameter alone.

Lidar Display Parameters

The parameters below allow specifying options for working with Lidar data.

- **LIDAR_DRAW_MODE** - specifies how points in a Lidar point cloud layer should be drawn. The following values are supported:
  - **COLOR** - if the points have an associated RGB color, use that. Otherwise color by elevation.
  - **ELEV** - color by elevation of the point using the current elevation shader.
  - **INTENSITY** - color as a grayscale image by the intensity
  - **CLASS** - color by the point classification
  - **RETURN** - color by the return number
  - **HEIGHT_ABOVE_GROUND** - color by the height above ground
  - **POINT_SOURCE_ID** - color by the point source ID
  - **POINT_INDEX** - color by the index of the point in the cloud
  - **RETURN_HEIGHT_DELTA** - color by the difference in height between first and last return
  - **CIR** - color as color infrared if NIR band present
  - **NDVI** - color by calculated NDVI value if NIR band present
  - **NDWI** - color by calculated NDWI value if NIR band present
  - **DENSITY** - color by point density
  - **WITHHELD** - color by withheld flag
  - **OVERLAP** - color by overlap flag
  - **KEY_POINT** - color by key point flag

- **LIDAR_POINT_SIZE** - specifies how large the points in a Lidar point cloud are drawn. The default value of **0** will automatically scale the points to be larger as you zoom in on them. Specify a fixed number to always draw them at a particular size in pixels.

- **LIDAR_DRAW_QUALITY** - specifies the quality setting (**0**-**100**) for drawing the Lidar point cloud. Larger values draw a larger fraction of the points when zoomed out, but the draw will take longer to complete.

- **LIDAR_FILTER** - specifies a comma-separated list of Lidar class numbers to enable or disable for load. Provide a minus sign (**-**) to remove the type from the filter rather than add it. The filter starts off with nothing in it if you provide a LIDAR_FILTER string, but you can add **ALL** to enable everything or **NONE** to clear the filter, then add or remove stuff after that. For example, to specify a class filter with only types 2 and 3 enabled, use `LIDAR_FILTER="NONE,2,3"`. To get one with everything but classes 2 and 3, use `LIDAR_FILTER="ALL,-2,-3"`.

- **LIDAR_RETURN_FILTER** - specifies a comma-separated list of Lidar return types to enable or disable for load. Provide a minus sign (**-**) to remove the type from the filter rather than add it. The filter starts off with loading everything, but you can add **ALL** to enable everything or **NONE** to clear the filter, then add or remove stuff after that. For example, to specify a return filter with only unknown and first returns, use `LIDAR_RETURN_FILTER="NONE,0,1"`. To get one with everything but the first return, use `LIDAR_RETURN_ FILTER="ALL,-1"`. The numeric values have the following meanings:

    - **0** - Unknown Returns
    - **1** - First Return
    - **2** - Second Return
    - **3** - Last Return
    - **4** - Single Return
    - **5** - First of Many Returns
    - **6** - Second of Many Returns
    - **7** - Third of Many Returns
    - **8** - Last of Many Returns

- **LIDAR_USE_INTEN_FOR_NIR** - specifies that we should treat the intensity value Lidar points as the NIR (near infrared) value if the point cloud has RGB colors but no NIR values of its own. Use LIDAR_USE_ INTEN_FOR_NIR=**YES** to enable.

Layer Rectification/ Control Points

The parameters below allow defining a series of control points and rectification parameters for setting up a coordinate mapping from pixel space to real-world projection coordinates for the layer.

- **GCP** - specifies a single ground control point for use in rectifying a file. The GCP record consists of 5 comma-delimited fields, *the control point name, the pixel X and Y coordinates, and the corresponding ground X and Y coordinates*. A separate GCP parameter and value should be used for each control point used in the rectification. As an alternative, the GCP_FILENAME parameter (see below) can be used instead.

- **GCP_FILENAME** - specifies the name of a control point file used to rectify the file being imported.

- **GCP_PROJ_NAME** - specifies the name of the projection that the ground control points are provided in. This name must have been defined with a prior DEFINE_PROJ command. Use this if you want to specify control points in a projection other than what you want to define as the native projection for the file. Note that you must also explicitly specify the name projection of the file using either the PROJ, PROJ_NAME, PROJ_ EPSG_CODE or PROJ_FILENAME parameters.

- **TRANSFORM_FILENAME** - specifies the name of a control point file used to transform the coordinates of the imported file. This is different than the GCP_FILENAME in that the file defines a mapping of world coordinates to a new set of world coordinates rather than pixel coordinates to world coordinates. Each line should be of the format: *x_orig,y_orig,x_new_y_new*

- **GCP_PROJ_FILENAME** - specifies the name of the projection (.prj) file that contains the projection definition for the projection that the ground control points are provided in. Use this if you want to specify control points in a projection other than what you want to define as the native projection for the file. Note that you must also explicitly specify the name projection of the file using either the PROJ, PROJ_NAME, PROJ_EPSG_ CODE or PROJ_FILENAME parameters.

- **GCP_PROJ_EPSG_CODE** - specifies the EPSG code of the projection that the ground control points are provided in. Use this if you want to specify control points in a projection other than what you want to define as the native projection for the file. Note that you must also explicitly specify the name projection of the file using either the PROJ, PROJ_NAME, PROJ_EPSG_CODE or PROJ_FILENAME parameters.

- **RECTIFY** - specifies the rectification method to use for rectifying this file. Valid value are **LINEAR**, **HELMERT**, **AFFINE**, **POLYNOMIAL**, and **TRIANGULATION**. If you do not specify a rectification type but do provide at least two ground control points, the best rectification method will automatically be chosen based on the number of control points specified.

- **RECTIFY_4_POINT_POLY_ONLY** - specifies that if RECTIFY=POLYNOMIAL is used to specify the rectification method, the polynomial will always be a 1st degree polynomial and won't switch automatically to a 2nd degree polynomial at 6 or more points. By default, the 2nd degree polynomial will automatically be used

IMPORT_ARCHIVE

The IMPORT_ARCHIVE command imports a data file from a .tar.gz archive for later use. The only time you should ever need to use the IMPORT_ARCHIVE command is when you only want to load some of the data inside a .tar.gz

archive. For the typical case of just loading everything in an archive, use the IMPORT command with AUTO as the value for the TYPE parameter. The following parameters are supported by the command:

- **ARCHIVE_FILENAME** - full path to the archive file to load the data from
- **FILENAME** - filename to load from the archive. You can include wildcard characters like '*' and '?' in the value to match on multiple files.

> NOTE: All other parameters that are supported by the IMPORT command are also supported by this command.

IMPORT_ASCII

The IMPORT_ASCII command imports data from a generic ASCII text file for later use. The following parameters are supported by the command. In addition, all of the option parameters for the IMPORT command are also supported for this command.

- **FILENAME** - full path to file to load the data from
- **TYPE** - type of import that we're doing
    - **POINT_ONLY** - all lines with coordinate data will result in a new point object begin created
    - **POINT_AND_LINE** - both point and line features (and optionally areas) will be created from coordinate data in the file. Line features will be created when coordinate data lines are back to back in the file. All individual coordinate lines will result in a point object being created
    - **AREA_ONLY** - only closed area features will be created from the sequences of coordinates.
    - **ELEVATION** - all lines in the file with 3 coordinate values (x,y, and elevation) will be used to create an elevation grid. The data will be triangulated and gridded automatically, resulting in a fully usable elevation grid that can be exported to any of the supported elevation data formats.
    - **LIDAR** - all 3D points in the file are added to a Lidar point cloud. You can load a .xyzi file to also add the intensity. Use the LIDAR_CLASS parameter to specify a numeric classification to apply to all points, like LIDAR_CLASS=2 to assign as ground shot points.
    - **DIST_BEARING** - all lines contain a distance and bearing from some other point location provided using the START_POS parameter. This will create point features. You should also provide a COORD_ORDER parameter with a custom definition for the column locations. The distances should be in meters and the bearings in degrees relative to north.
    - **DIST_BEARING_LINE** - all lines contain a distance and bearing for a segment of a line. The line starts at the point location provided using the START_POS parameter. This will create a single line feature. You should also provide a COORD_ORDER parameter with a custom definition for the column locations. The distances should be in meters and the bearings in degrees relative to north.
    - **DIST_BEARING_SEGS** - all lines contain a distance and bearing for a segment of a line. For each line in the file, a line will be created that starts at the point location provided using the START_POS parameter. You should also provide a COORD_ORDER parameter with a custom definition for the column locations. The distances should be in meters and the bearings in degrees relative to north.
    - **COORD_DELIM** - specifies the delimeter between coordinates in coordinate lines
    - **AUTO** - automatically detect the delimeter type (default)
    - **WHITESPACE** - coordinates are separated with one or more space and/or tab characters
    - **COMMA** - coordinates are separated by commas
    - **SEMICOLON** - coordinates are separated by semicolons
    - **TAB** - coordinates are separated by tabs
- **COORD_FORMAT** - specifies the format of the coordinate values. The default is DECIMAL.
    - **DECIMAL** - standard numerical value. No extra values are packed into one.
    - **DDMMSS** - degree coordinates are stored as the number of degrees times 10,000 plus the minutes times 100 plus the seconds. So for example 35 deg 15 min and 12.3 seconds would look like 351512.3.
    - **DDMM** - degree coordinates are stored as the number of degrees times 100 plus the minutes. So for example 35 deg 15.2 min would look like 3515.2.
    - **DD_MMSS** - degree coordinates are stored as DD.MMSS (i.e. degrees + (minutes / 100) + (seconds / 10000). So for example 35 deg 15 min and 12.3 seconds would look like 35.15123.
    - **ECEF** - the XYZ coordinates represent ECEF (Earth-Centered Earth-Fixed) coordinates. They actual values are the same as the DECIMAL format. The ECEF coordinates will be converted to lat/lon degrees using the ellipsoid implied by the projection datum, then converted to whatever projection is specified.
- **COORD_ORDER** specifies the order of the coordinates in coordinate lines
    - **X_FIRST** - x coordinates (i.e. easting or longitude) come first, followed by y coordinates (i.e. northing or latitude) (default)
    - **Y_FIRST** - y coordinates (i.e. northing or latitude) come first, followed by x coordinates (i.e. easting or longitude)
    - **WKT** - coordinate string in WKT (well-known-text format). This allows single line representations of areas, lines, and points.
    - **MGRS** - MGRS (military grid reference system) coordinate string

- **CUSTOM** - specifies a custom column layer. The columns are specified as 1-based numbers (i.e. first column is 1, not 0). The values should be specified as "CUSTOM,x_col,y_col[,z_col][,time_col]", or for DIST_BEARING as "CUSTOM,dist_col,bearing_col[,z_col]". So for example if the X/longitude is in column 3, the Y/latitude in column 4, and the Z in the first column, use "CUSTOM,3,4,1". If you need to specify that an optional colulmn isn't present, use -1 for that column. For example if you have an X,Y,time file, use "CUSTOM,1,2,-1,3".

- **COORD_PREFIX** - if present, this line is used to specify what special character sequence coordinate lines start with. For example, if the coordinate lines in the file started with the character sequence "XY,", you should use COORD_PREF="XY,". By default no coordinate prefix is assumed.

- **INC_COORD_LINE_ATTRS** - set the value of this parameter to **YES** if you wish to use any leftover text at the end of coordinate lines as attributes for the feature the coordinates are in. This could be useful if elevation data is present at the end of the lines. By default, the value of this attribute is **NO**.

- **INC_ELEV_COORDS** - this parameter controls whether or not the value right after the 2nd coordinate column (if there is one) will be treated as an elevation value. Use INC_ELEV_COORDS=**YES** or INC_ELEV_COORDS=**NO** to enable (the default) or disable this behavior.

- **VOID_ELEV** - specifies what value is treated as a no-data (void) elevation and ignored for the purposes of gridding or adding an elevation attribute. If not provided the default invalid value of -**99999.0** is used.

- **COL_HEADERS** - controls whether or not the first line of the file should be used as column headers for coordinate line attributes later in the file. Setting this to **YES** is useful for things like CSV files with column headers in the first row, otherwise set it to **NO** (the default).

- **SKIP_COLUMNS** - specifies the number of columns (fields) to skip at the start of a coordinate line before trying to read the coordinates. For example, if the X and Y coordinates of a line were in the 3rd and 4th columns of the coordinate line, you'd use a value of SKIP_COLUMNS=2. The default value is **0**, meaning that coordinates must be in the first two columns.

- **SKIP_ROWS** - specifies the number of rows to skip at the start of a file before trying to read any data. For example, if your file has a fixed header of 20 lines, you would use SKIP_ROWS=20 to skip those header rows.

- **COORD_PAIRS_PER_ROW** - specifies the number of coordinate pairs (XY + optional Z and time) that are on each line of the file. If specified the value must be at least **1**. Use this to load line or area features from files that have 2 or more coordinate pairs on each line of the file. The pairs are assumed to be sequential, so if you have 3 XYZ coordinate pairs on a line and the first 2 fields are attributes, use SKIP_COLUMNS=2 COORD_PAIRS_PER_ROW=3 to get a triangle from each line in the format attr1,attr2,x1,y1,z1,x2,y2,z2,x3,y3,z3.

- **BREAK_COL_IDX** - specifies the 1-based index of the column to break features at if the value in that column changes.

- **BREAK_COL_PEN_UP** - specifies that the "break on change column" from the BREAK_COL_IDX parameter is actually a pen up/down field and new features should be started when a 1 is encountered in the field. Use BREAK_COL_PEN_UP=**YES** to enable.

- **CREATE_AREAS_FROM_LINES** - controls whether or not area features will be created from closed line features (first and last point the same) if no CLOSED attribute was specifically provided for the feature.

- **COORD_OFFSET** - specifies the offset to apply to any coordinates read in from the file. This offset will be added to each coordinate read in from the file. The offset should be specified as a comma-delimited list of the X, Y, and Z offsets, such as COORD_OFFSET=100000.0,200000.0,0.0

- **COORD_SCALE** - specifies the scale factor to apply to any coordinates read in from the file. Each coordinate will be multiplied by these scale factor after being read in from the file. The scale factors should be specified as a comma-delimited list of the X, Y, and Z scale factors, such as COORD_SCALE=0.1,0.1,1.0

- **NO_DATA_DIST_MULT** - specifies how far from an actual data point a grid cell has to be before it is treated as a no data value. This number is given as a multiple of the diagonal size of a single grid cell as nominally determined by the gridding algorithm or specified with the SPATIAL_RES parameter. A value of **0** means that all points should be considered as valid.

- **SPATIAL_RES** - specifies spatial resolution to use when generating an elevation grid from the data. Defaults to a good value for maintaining the full spatial resolution of the provided point data if not specified. Should be formatted as x_resolution,y_resolution. The units are the units of the projection specified for the file. For example, if UTM with meter units was the file projection and you wanted to export at 30 meter spacing, the parameter/value pair would look like SPATIAL_RES=30.0,30.0.

- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will automatically be converted to the curent view/export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use SPATIAL_RES_METERS=2.0, or to do an export at 1.0 meters in X by 1.5 meters in Y, use SPATIAL_RES_METERS="1.0,1.5".

Distance-Bearing Type Parameters

The following parameters are applicable when loading a file set as TYPE=DIST_BEARING , TYPE=DIST_BEARING_ LINE or TYPE=DIST_BEARING_SEGS

- **START_POS** - specifies the start position for distance-bearing files. The coordinates must be given in the coordinate system of the layer. For example, if UTM if the current projection, you might specify and east-ing/northing as follows: START_POS="480000,4310000". You can also specify START_POS="SELECTED" to use the location of a point feature selected with the Digitizer Tool as the position.
- **DIST_BEARING_REL_TO**- indicates the basis for the bearing angle. Valid values are: **TRUE_NORTH**, **MAG_NORTH**, and **GRID_NORTH**. If this parameter is not specified, the default is TRUE_NORTH.
- **DIST_BEARING_UNIT**- specifies the linear unit for the distance values. Both the full name and abbre-viation are valid. If this parameter is not specified, the default is **meters**.

SAMPLE

```
IMPORT_ASCII FILENAME="C:\data\ASCII Files\usvi_landmark.asc" \
     TYPE=POINT_AND_LINE COORD_DELIM=AUTO COORD_ORDER=X_FIRST \
     COORD_PREFIX="XY,"INC_COORD_LINE_ATTRS=NO
```

IMPORT_CLOUD

The IMPORT_CLOUD command imports data from a cloud dataset, including Amazon S3 account. The following parameters are supported by the command.

- **FILENAME** - Name of the file when it is downloaded
- **CLOUD_TYPE** - Cloud type, currently only AWS is supported "**Amazon's AWS S3**"
- **CLOUD_KEY1** - first access key, for AWS S3 this is the Public Key
- **CLOUD_KEY2** - second access key, for AWS S3 this is the Private Key
- **CLOUD_FOLDER** - folder where file exists in the cloud, for AWS S3 this is the bucket
- **CLOUD_LOCATION** - location where folder exists, for AWS S3 this is region
- **CLOUD_FILE** - the name of the file as it exists in the cloud.

EXAMPLE

```
IMPORT_CLOUD FILENAME="C:\data\bilbo.shp" TYPE="SHAPEFILE" \
ELEV_UNITS="METERS" LABEL_FIELD_FORCE_OVERWRITE="YES" LABEL_FIELD_SEP="0x20" LABEL_FIELD="NAME"
\
CODE_PAGE="0" CLOUD_TYPE="Amazon's AWS S3" CLOUD_KEY1="AAAAAAAAAAAAAAAAAAAA" \
CLOUD_KEY2="AAAA+bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb" \
CLOUD_FOLDER="bmg-billingbucket" CLOUD_LOCATION="us-east-1" CLOUD_FILE="bilbo.shp"
```

IMPORT_DIR_TREE

The IMPORT_DIR_TREE command imports all of the data files in a given directory tree that match a list of filename masks. The following parameters are supported by the command. In addition, all of the option parameters for the IMPORT command are also supported for this command.

- **DIRECTORY** - full path to root of directory tree to import files from.
- **FILENAME_MASKS** - space-separated list of filename masks to import. If no value is provided then all files which are of recognized types will be imported.
- **RECURSE_DIR** - specifies whether the search for matching files will recurse in to sub-folders. The default is RECURSE_DIR=**YES**. Use RECURSE_DIR=**NO** to only search in the specified folder.

SAMPLE

```
IMPORT_DIR_TREE DIRECTORY="C:\TEMP\EXPORT TEST" FILENAME_MASKS="*.OPT *.GMP"
```

DEFINE_SDB_CONNECTION

The DEFINE_SDB_CONNECTION allows the user to define a connection and use that definition in the script. This is required for enterprise spatial databases that require a defined connection, but is not needed for exporting to a file-based spatial database such as Esri Personal Geodatabase or Spatialite/SQLite.

Using the SAVE_CONNECTION parameter will cause this definintion to be stored with the connections defined using the Connection Manager.

- **SDB_CONNECTION_NAME** - The name of the connection. This is used to identify the connection in a subsequent EXPORT_VECTOR or IMPORT_SPATIAL_DB command.
- **TYPE** - Spatial DB Type Name, one of:
  - **ESRI_ARCSDE** - Esri ArcSDE Geodatabase
  - **MSSQLSERVER** - Micorsoft SQL Server Spatial
  - **MYSQL** - MySQL Spatial
  - **ORACLE** - Oracle Spatial Database
  - **POSTGIS** - PostGIS/PostgreSQL
- **SDB_SERVER** - the server where the database is located.
- **SDB_PORT** - the required server port number. In an ArcSDE geodatabase connection, this is also known as the Service parameter.
- **SDB_DATABASE_NAME** - The database name. This is optional when defining an Esri ArcSDE geodatabase connection. Whether or not it is needed depends on your particular installation.
- **SDB_USER_NAME** - The user name that will be used to access the database
- **SDB_PASSWORD** - The password in plain text
- **SDB_SAVE_USER_AND_PASSWORD** - If the SAVE_CONNECTION=YES parameter is used, this indicates whether or not to save the user name and password with the connection definition.
- **SDB_USE_OS_AUTH** - Use current OS login credentials to connect to the database. Specify SDB_USE_OS_AUTH instead of a user name and password. Supported only for Microsoft SQL Server and Oracle Spatial Database connections.
- **SAVE_CONNECTION** - Boolean value that indicates whether or not this definition should be saved in the registry (if it does not already exist). If value is **NO**, then this definition will only be used in this script. Default is NO.
- **OVERWRITE_EXISTING** - Boolean value that indicates whether or not the definition in the script should replace the existing definition. Default will be **NO**.

SAMPLE

```
DEFINE_SDB_CONNECTION TYPE="POSTGIS" SDB_CONNECTION_NAME="PostGIS" \
 SDB_SERVER="myserver" SDB_PORT="5432" SDB_DATABASE_NAME="mydb" \
 SDB_SAVE_USER_AND_PASSWORD="YES" SDB_USER_NAME="pguser" \
 SDB_PASSWORD="pgpassword" SAVE_CONNECTION=YES
```

IMPORT_SPATIAL_DB

The IMPORT_SPATIAL_DB command allows the user to import spatial data from a spatial database. The database can be either a file-based spatial database or a connection-based spatial database.:

- **TYPE** - *File-Based Spatial Databases* (Using these TYPE values requires that the SDB_CONNECTION_FILE parameter also be specified to identify the spatial database to be used.)

- **SPATIALITE** - Spatialite/SQLite
- **FILE_GDB** - Esri File Geodatabase
- **ESRI_PGEO** - Esri Personal Geodatabase
- **TYPE** - *Connection-Based Spatial Databases* (Using these type values requires that the SDB_ CONNECTION_NAME parameter also be specified to identify the connection to be used.)
  - **ESRI_ARCSDE** - Esri ArcSDE Geodatabase
  - **ESRI_XML_WORKSPACE** - Esri XML Workspace
  - **MSSQLSERVER** - Micorsoft SQL Server Spatial
  - **MYSQL** - MySQL Spatial
  - **ORACLE** - Oracle Spatial Database
  - **POSTGIS** - PostGIS/PostgreSQL
- **SDB_CONNECTION_FILE** - When importing from a file-based spatial database, use this parameter to specify the full path to the database file. If importing from an Esri File Geodatabase, this parameter must contain the directory containing the geodatabase (typically ends in ".gdb" even though it is a directory).
- **SDB_CONNECTION_NAME** - The name of the connection to be used to access a connection-based spatial database. Connections can be defined in the script using a DEFINE_SDB_CONNECTION command, or by using the Connection Manager in the Global Manager user interface. All of the connections defined in the Connection Manager are available for use in a script.
- **SDB_TABLE_NAME** - Each IMPORT_SPATIAL_DB command will import a single database table. Use this parameter to specify the name of the database table to be imported.
- **SDB_IMPORT_BOUNDS** - specifies the bounds to be used when importing data from a spatial database. The parameter value is a bounding rectangle in the same projection as the import database, in the form: "*<minX>,<minY>,<maxX>,<maxY>*".
- **LAT_LON_BOUNDS** - specifies the bounds to import in latitude/longitude degrees relative to the WGS84 datum. There should be 4 values in a comma-delimited list following the parameter name. The values should be in order of westmost longitude, southernmost latitude, easternmost longitude, northernmost latitude.
- **LAYER_BOUNDS** - specifies that the import should use the bounds of the loaded layer(s) with the given filename. For example, to import to the bounds of the file "c:\test.tif", you would use `LAYER_BOUNDS-S="c:\test.tif"`. Keep in mind that the file must be currently loaded.
- **LAYER_BOUNDS_EXPAND** - specifies that the operation should expand the used LAYER_BOUNDS bounding box by some amount. The amount to expand the bounding rectangle by should be specified in the current global projection. For example, if you have a UTM/meters projection active and want to expand the bounds retrieved from the LAYER_BOUNDS parameter by 100 meters on the left and right, and 50 meters on the top and bottom, you could use `LAYER_BOUNDS_EXPAND="100.0,50.0"`. You can also specify a single value to apply to all 4 sides, or supply 4 separate values in the order *left,top,right,bottom*.

SAMPLES

```
IMPORT_SPATIAL_DB TYPE="POSTGIS" SDB_CONNECTION_NAME="PostGIS" \
 SDB_TABLE_NAME="public.canada" SDB_IMPORT_BOUNDS="-126.821609,26.773888,-106.597575,50.302504"
```

```
IMPORT_SPATIAL_DB TYPE="ESRI_XML_WORKSPACE" SDB_CONNECTION_FILE="filename" \
 SDB_TABLE_NAME="tablename"
```

IMPORT_OSM_TILE

The IMPORT_OSM_TILE command imports a tiled online layer using the OSM, TMS, Google Maps, or Bing Maps tile schema. The following parameters are supported by the command:

- NOTE: All style parameters that are supported by the IMPORT command are also supported by this command.

- **OSM_BASE_URL** - URL to base of tile source. Can include custom URL variables like *%z*, *%x*, *%y*, or *%quad* (for Bing-style naming) for defining exactly how the request URL should look. See the add online source dialog for more information and a sample of a custom URL.
- **OSM_DESC** - description to use for the source
- **OSM_FILE_EXT** - file extension for tiles, like PNG, JPG, or GMG (for terrain).
- **OSM_NUM_ZOOM_LEVELS** - specifies the number of the maximum zoom level for the source. Note if this is a built-in source you don't need to provide this, just leave it off and the default will be used.
- **TILE_SIZE** - specifies the size in pixels of each tile. For example if the source uses 512x512 tiles, add TILE_SIZE=512. The default is 256.
- **DETAIL_MULT** - specifies the detail scale to use when deciding while zoom level to get for the source. The calculated draw/export resolution is divided by this value to get the resolution to access the data at. For example, a value of DETAIL_MULT=0.5 means the source will be displayed from twice the detail it normally would, while DETAIL_MULT=2 would pull at half the resolution (i.e. much faster access).
- **LEVEL0_TILECOLS** - Number of tile columns across the lowest zoom level. By default this is 1 for a single tile covering the entire world.
- **LEVEL0_TILEROWS** - Number of tile rows at the lowest zoom level. By default this is 1 for a single tile covering the entire world.
- **SAMPLE_TYPE** - Specifies the format of samples in BIL files for a terrain-based layer. Typical values will be F32 for 32-bit floats, S32 for 32-bit signed integer, or S16 for 16-bit signed integer.

Specify Tiling Type

By default the OSM tile naming schema is assumed, but you can use the parameters below to specify Google Maps or TMS tiling. If you specify a full custom URL in OSM_BASE_URL then the tiling schema doesn't matter as much since the URL defines the naming.

- **SOURCE_TYPE** - specifies the tile schema for the online source. This is supported in v16.2.4 and later and takes precedence over the deprecated OSM_IS_GOOGLE_MAPS and OSM_IS_TMS parameters. The acceptable values are:
  - **GMAP** - Google Maps tiles
  - **OSM** - OSM tiles
  - **TMS** - TMS tiles
  - **VWORLD_DEM** - VWorld DEM terrain in BIL tiles, tiles increase from bottom up
- **OSM_IS_GOOGLE_MAPS** - add OSM_IS_GOOGLE_MAPS=**YES** to indicate the source uses the Google Maps tiling scheme
- **OSM_IS_TMS** - add OSM_IS_TMS=**YES** to indicate the source uses the TMS tiling scheme

Specify Bounds for Layer

Use the parameters below to define the bounding box to import from the source:

- **ADDRESS** - address to download data near. Use along with RADIUS to specify the bounds with an address and radius rather than a specific bounding box.
- **RADIUS** - radius in kilometers around ADDRESS to search.
- **LAT_LON_BOUNDS** - specifies the bounds to import in latitude/longitude degrees relative to the WGS84 datum. There should be 4 values in a comma-delimited list following the parameter name. The values should be in order of westmost longitude, southernmost latitude, easternmost longitude, northernmost latitude.
- **LAYER_BOUNDS** - specifies that the import should use the bounds of the loaded layer(s) with the given filename. For example, to import to the bounds of the file "c:\test.tif", you would use `LAYER_BOUNDS-S="c:\test.tif"`. Keep in mind that the file must be currently loaded.
- **LAYER_BOUNDS_EXPAND** - specifies that the operation should expand the used LAYER_BOUNDS bounding box by some amount. The amount to expand the bounding rectangle by should be specified in the current global projection. For example, if you have a UTM/meters projection active and want to expand the bounds retrieved from the LAYER_BOUNDS parameter by 100 meters on the left and right, and 50 meters on the top and bottom, you could use `LAYER_BOUNDS_EXPAND="100.0,50.0"`. You can also specify a single value to apply to all 4 sides, or supply 4 separate values in the order left,top,right,bottom.

SAMPLE

Here is an example of an IMPORT_OSM_TILE command that pulls in MapQuest OpenStreetMap data within 5 km of Blue Marble's headquarters:

```
 IMPORT_OSM_TILE OSM_BASE_URL="http://otile1.mqcdn.com/tiles/1.0.0/osm/" OSM_DESC="MapQuest
OpenStreetMap Worldwide Street Maps" \ \
 OSM_FILE_EXT="png" OSM_NUM_ZOOM_LEVELS="19" ADDRESS="77 Water St, HALLOWELL, ME" RADIUS="5" \
 CENTER_LABEL="Blue Marble Geographics" CENTER_LABEL_POS="-69.7908786,44.2859022"
```

IMPORT_WMS

The IMPORT_WMS command imports a chunk of WMS or WMTS (tiled WMS) data, such as satellite imagery or topo-graphic maps. The following parameters are supported by the command:

- **SOURCE_DESC** - text description of source. Should match name from the online source dialog. Used to match to a source in the source list if no match based on URL could be found.
- **WMS_SERVER_URL** - URL to WMS server GetCapabilites
- **WMS_SERVICE** - service name to use, typically **WMS**
- **WMS_LAYER** - name of WMS layer to load
- **WMS_IS_TILED** - specifies that the server is a WMTS (tiled WMS) service. Use WMS_IS_TILED=**YES** to enable.
- **ADDRESS** - address to download data near. Use along with RADIUS to specify the bounds with an address and radius rather than a specific bounding box.
- **RADIUS** - radius in kilometers around ADDRESS to search.
- **LAT_LON_BOUNDS** - specifies the bounds to import in latitude/longitude degrees. There should be 4 val-ues in a comma-delimited list following the parameter name. The values should be in order of *west-most longitude*, *southern-most latitude*, *eastern-most longitude*, *northern-most latitude*.
- **LAYER_BOUNDS** - specifies that the import should use the bounds of the loaded layer(s) with the given filename. For example, to import to the bounds of the file "c:\test.tif", you would use `LAYER_BOUNDS-S="c:\test.tif"`. Keep in mind that the file must be currently loaded.
- **LAYER_BOUNDS_EXPAND** - specifies that the operation should expand the used LAYER_BOUNDS bound-ing box by some amount. The amount to expand the bounding rectangle by should be specified in the cur-rent global projection. For example, if you have a UTM/meters projection active and want to expand the bounds retrieved from the LAYER_BOUNDS parameter by 100 meters on the left and right, and 50 meters on the top and bottom, you could use `LAYER_BOUNDS_EXPAND="100.0,50.0"`. You can also specify a single value to apply to all 4 sides, or supply 4 separate values in the order left,top,right,bottom.
- **DETAIL_MULT** - specifies the detail scale to use when deciding while zoom level to get for the source. The calculated draw/export resolution is divided by this value to get the resolution to access the data at. For example, a value of `DETAIL_MULT=0.5` means the source will be displayed from twice the detail it normally would, while `DETAIL_MULT=2` would pull at half the resolution (i.e. much faster access).
- NOTE: All style parameters that are supported by the IMPORT command are also supported by this com-mand.

SAMPLE

Here is an example of an IMPORT_WMS command that pulls in NAIP imagery within 5 km of Blue Marble's headquar-ters:

```
 IMPORT_WMS WMS_SERVER_URL="http://isse.cr.usgs.gov/arcgis/services/Combined/USGS_EDC_Ortho_
NAIP/MapServer/WMSServer"\ \
 WMS_SERVICE="WMS" WMS_LAYER="0" ADDRESS="397 WATER ST, GARDINER, ME" RADIUS="5" \
 LAYER_DESC="NAIP Color Imagery for US (1m Resolution)"
```

Layer Management

COPY_LAYER_FILES

The COPY_LAYER_FILES command copies the base files for one or more layers to a new folder on disk. Support is included for maintaining folder structures if a BASE_DIR parameter is provided. If you specify layers that were loaded from .zip or .tar.gz archives, the archive file itself will be copied and not the individual extracted files. If you specify a file with supporting files with the same base name (i.e *foo.tfw* and *foo.prj* with *foo.tif* loaded) they will also be copied.

The following parameters are used by the COPY_LAYER_FILES command.

- **FILENAME** - filename or description of layer(s) to copy the files for. This can include * and ? wildcard char-acters. If you leave the FILENAME parameter off then all loaded layers will have their files copied, which is the same behavior as using `FILENAME="*"`. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`.
- **TARGET_DIR** - specifies the folder where the files will be copied to. If no BASE_DIR parameter is provided, the files will all be copied directly to the specified folder.
- **BASE_DIR** - specifies a starting string for the layer files being copied beyond which everything should be treated as relative. For exampling if copying a file at "*c:\data\my_dems\colorado\denver.dem*" to a TARGET_DIR of "*c:\new_dems*" with a BASE_DIR value of "*c:\data\my_dems*" you would get "*c:\new_dems\colorado\denver.dem*" as the new filename.
- **OVERWRITE_EXISTING** - specifies that existing files should be overwritten. The default is OVERWRITE_EXISTING=**YES**, so use OVERWRITE_EXISTING=**NO** to skip files that already exist in the destination loc-ation.

SAMPLE

```
GLOBAL_MAPPER_SCRIPT VERSION="1.00"
/* Copy all loaded layer files to a new folder */
COPY_LAYER_FILES TARGET_DIR="C:\NEW_DEMS" BASE_DIR="c:\data\my_dems" OVERWRITE_EXISTING=YES
```

GENERATE_LAYER_BOUNDS

The GENERATE_LAYER_BOUNDS command create a new layer with a single bounding box area created from the bounding box of each loaded layer or a polygonal coverage of the valid data in the layer if specified by the BOUNDS_TYPE parameter:

- **LAYER_DESC** - specifies the description to use the for created layer
- **FILENAME** - filename of the layer to generate bounds for. If an empty value is passed in, all layers that were created by the script, such as those from a GENERATE_CONTOURS command, will have bounds created for. This parameter can be listed more than once to specify multiple input files, like `FILENAME-E="FILENAME_1" FILENAME="FILENAME_2"`. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value **'USER CREATED FEATURES'** to have the 'User Created Features' layers unloaded or **'SELECTED LAYERS'** to have any layers selected in the Control Center unloaded. You can also pass in the *full description* of the loaded layer to use in case you want to process a layer not loaded from a file. If you do not provide a FILENAME parameter then all loaded layers will have their bounds generated.
- **BOUNDS_TYPE** - specifies whether to create bounding box or polygon coverages. The following values are supported:
  - **BOUNDS** - (**default**) A bounding box in the native projection of the layer is created.
  - **POLYGON** - A coverage polygon is calculated covering the features/valid data in the layer
  - **RECT_ONLY** - This is the same as the BOUNDS value, except the bounding box is in the current projection rather than the native layer projection.
- **MAX_VERTEX_COUNT** - specifies the maximum number of vertices to include in a polygonal coverage. The **default** is MAX_VERTEX_COUNT=**32**. If the polygonal coverage has more than the specified maximum count the polygon will be simplified until it has less vertices than the specified count.
- **COVERAGE_SMOOTHING_FACTOR** - specifies a smoothing factor to use when creating polygonal coverage areas for vector/Lidar layers to control how tightly shrink-wrapped around the vector features the area is. The **default** value is **1.0**, but *any value greater than 0* is allowed, with larger values resulting in more smoothing.

SET_LAYER_OPTIONS

The SET_LAYER_OPTIONS command sets the display options for one of more loaded layers. These are the options that you would normally supply when importing a layer. The following parameters are supported by the command. In addition, all of the option parameters for the IMPORT command are also supported for this command.

- **FILENAME** - full path or description of loaded layer to set the options for. This can include wildcard characters like '*'. If you specify an empty filename then all layers that have no filename, like generated contours, will be matched. If you leave of the FILENAME parameter entirely or use a '*' wildcard (like FILENAME="*") then all loaded layers will be updated. If you want layers only in some groups, you can add <sub> to the FILENAME. For example, to match all layers in the layer group 'Line Group', use FILENAME="Line Group<sub>*". When running the script in the context of the main map view (including loading a workspace) you can also pass in the value 'USER CREATED FEATURES' to have the 'User Created Features' layer updated or 'SELECTED LAYERS' to have any layers selected in the Control Center.

Shared Import Parameters

These parameters are shared across import and layer options commands to set layer properties.

- **HIDDEN** - set to **YES** to cause this overlay to be hidden from view after it is loaded. The default is to show the overlay.
- **LAYER_DESC** - specifies a description to use for the layer when displaying it in the Overlay Control Center. This overrides the default description based on the filename or other information within the file.
- **LAYER_GROUP** - specifies the name of the group for the layer in the Control Center. To include multiple layers of grouping put the string <sub> in between levels. For example to make a group with 2 levels of nesting, use LAYER_GROUP="Top Level<sub>Next Level".
- **ALLOW_SELECTION** - set to **NO** to disable selection of features from this layer using either the Feature Info or Digitizer Tools.
- **ALLOW_EXPORT** - set to **NO** to disable export from this layer.
- **LOAD_FLAGS** - contains flags for any import options that you were prompted for when loading the file, such as if you have a .tif file that you were prompted to select as elevation or raster. Also things like the coverages and tile sets for VPF layers. To see how to set these if you are writing a script, load a file with the settings that you want in the main user interface and then save a workspace, then examine the IMPORT command in the .gmw file for that file and see how the LOAD_FLAGS were set.

- **METADATA_FILENAME** - specifies full path and filename of a file to display the contents of on the Metadata dialog for a layer. The file can be any simple displayable text format, including text and XML.
- **METADATA_URL** - specifies a URL to a displayable web file (including HTML web page or XML document) to show on the Metadata dialog for a layer.
- **CODE_PAGE**- specifies the code page to use when interpreting text from this layer. By default if the file doesn't specify a code page the current system code page will be used. Use the code page number, or the text UTF-8 (number 65001).
- **ALT_MODE** (vector only) - altitude mode specifies how the 3D viewer should interpret z-values in the vector features of an layer, relative to terrain. Altitude mode may also be set in an individual feature, in which case it overrides the layer setting. The following values are supported:
    - **UNSPECIFIED** - Altitude mode is determined by either the setting in the feature, or if unspecified, the setting in the 3D viewer
    - **ABSOLUTE** - treat z-values as absolute elevations, ignoring any terrain
    - **RELATIVE_TO_GROUND** - treat z-values as distances above the terrain
    - **RELATIVE_TO_SEA_FLOOR** - treat z-values as distances above the sea floor (currently implemented as RELATIVE_TO_GROUND)
    - **CLAMP_TO_GROUND** - ignore z-values, and clamp the feature to the terrain
    - **CLAMP_TO_SEA_FLOOR** - ignore z-values, and clamp the feature to the sea floor (currently implemented as CLAMP_TO_GROUND)
    - **DEPTH** - treat z-values as absolute depths, ignoring any terrain
- **ZOOM_DISPLAY** - specifies when the map should be displayed and when it should be hidden based on the display zoom scale. This command will be formatted as a name from the list, below followed by 2 numeric paramters. For example, use `ZOOM_DISPLAY="SCALE,25000,0"` to have a map display only when zoomed in below 1:25000 scale.
    - **ALWAYS** - always display the map. The numeric parameters are ignored.
    - **PERCENT** - display the map when the map bounding box is a certain percentage of the screen size. For example, use `ZOOM_DISPLAY="PERCENT,0.10,0"` to display the map when its bounding box is at least 10% of the screen size.
    - **PIXEL_SIZE** - display the map when each display pixel is less than some number of meters in size. For example, use `PIXEL_SIZE="SCALE,10,0"` to display the map when the current display resolution is 10 meters per pixel (or less/higher resolution).
    - **SCALE** - display the map when the current display is at or below a certain scale. For example, use `ZOOM_DISPLAY="SCALE,25000,0"` to display the map when the current draw scale is at or below 1:25000.
    - **SCALE_RANGE** - display the map when the current display is below a range of scale value. For example, use `ZOOM_DISPLAY="SCALE_RANGE,25000,100000"` to display the map when the current draw scale is between 1:25000 and 1:100000.
- **PROJ** - special [Projection Specification](#) type of parameter that specifies the projection to use for the file. This will override any projection information stored in the file.
- *PROJ_NAME* (DEPRECATED use PROJ instead) - specifies the name of the projection to use for this file (this will override any projection information stored in the file). This name must have been defined with a prior DEFINE_PROJ command.
- *PROJ_FILENAME* (DEPRECATED use PROJ instead)- specifies the name of the projection (.prj) file to use for this file (this will override any projection information stored in the file).
- *PROJ_EPSG_CODE* (DEPRECATED use PROJ instead) - specifies the numeric EPSG projection code that defines the projection for this file (this will override any projection information stored in the file). For example, use PROJ_EPSG_CODE=26715 to define a UTM zone 15 projection with NAD27 as the datum and meters as the units.
- **PROMPT_IF_PROJ_UNKNOWN** - set to **NO** if you don't want the user to be prompted to select a projection if the projection of the file cannot be automatically determined.
- **USE_DEFAULT_PROJ** - specifies that if no projection can be automatically determined for a layer that the default projection selection should be used rather than prompting the user. Use USE_DEFAULT_PROJ=**YES** to enable. The default projection uses the first valid option from the following, including a check for linear versus angular numeric ranges:
    - Projection of any files loaded from the same folder
    - Last projection user selected on a projection dialog in this session
    - Current view projection
    - Projection from default.prj in global_mapper.exe path
    - Projection from default.prj in User Settings File path

- Last projection user selected on a projection dialog in previous session of GM
- Default UTM/15N/NAD83 projection

- **USE_DEFAULT_POS** - specifies that if no position data for a raster layer can be automatically determined that a default position should be chosen so that it displays. Use USE_DEFAULT_POS=**YES** to enable.

- **PICTURE_POS** - specifies that the image should be loaded as a 'picture point' that displays the image when you select the point with the Feature Info Tool. The value should contain the *X and Y coordinates* (in the projection specified for the layer). For example to place the value at 30N 95W with the projection set as PROJ_EPSG_CODE=4326 you can use `PICTURE_POS="-95.0,30.0"`.

- **LOAD_HIDDEN_PDF_LAYERS** - for PDF import, specifies that if no layer prompt is provided that hidden layers should be loaded automatically. Use LOAD_HIDDEN_PDF_LAYERS=**YES** to enable.

Elevation Parameters

Parameters for display and interpretation of elevation values in terrain layers. See also Raster Parameters below for additional shared parameters.

- **ELEV_FIELD** - specifies the name of the attribute field to use as the elevation value for the vector features in a file

- **ELEV_UNITS** - specify elevation units to use for this file if it contains gridded elevation data and also for vector feature elevations that don't have a unit embedded in the elevation value. Valid values are as follows:

  - **FEET** - elevations in US feet
  - **DECIFEET** - elevations in 10ths of US feet
  - **METERS** - elevations in meters
  - **DECIMETERS** - elevations in 10ths of meters
  - **CENTIMETERS** - elevations in centimeters

- **ELEV_OFFSET** (elevation only) - specifies the offset in meters to apply to each elevation value in the layer. This allows you to vertically shift a layer to match other layers.

- **ELEV_POWER** (elevation only) - specifies the power value to apply to each elevation value in the layer. For example a value of **2.0** would square each elevation value before applying a scale and adding the offset. Default to **1.0** (no power).

- **ELEV_SCALE** (elevation only) - specifies the scale value to apply to each elevation value in the layer. This allows you to vertically scale a layer to match other layers. Default to **1.0** (no scaling).

- **MIN_ELEV** (elevation only) - specifies the minimum elevation (meters) to treat as valid when rendering this layer. Any elevations below this value will be treated as invalid and not be drawn or exported.

- **MAX_ELEV** (elevation only) - specifies the maximum elevation (meters) to treat as valid when rendering this layer. Any elevations above this value will be treated as invalid and not be drawn or exported.

- **CLAMP_ELEVS** (elevation only) - if a MIN_ELEV and /or MAX_ELEV value is specified, setting this to **YES** will make any valid elevation values outside of the specified range be clamped to the new range value rather than treated as invalid.

- **VOID_ELEV** (elevation only) - specifies the elevation (meters) to replace any void areas in the layer with. If not specified, the void areas will be transparent.

- **SHADER_NAME** (elevation only) - this sets the name of the shader to use when rendering the gridded elevation data for this layer. Use this to override use of the shared default shader just for this layer. This must be one of the names displayed in the shader drop down in Global Mapper, such as **"Atlas Shader"** or **"Global Shader"** or the name of a custom shader.

Raster Parameters

Parameters for display of imagery. Some of the below parameters are also supported for elevation layers.

- **SAMPLING_METHOD** (elevation and raster only) - specifies the sampling method to use when res-ampling this layer.

  The following values are supported

  - **NEAREST_NEIGHBOR** - use the nearest neighbor sampling method
  - **BILINEAR** - use bilinear interpolation
  - **BICUBIC** - use bicubic interpolation
  - **BOX_2X2** - use a 2x2 box average
  - **BOX_3X3** - use a 3x3 box average
  - **BOX_4X4** - use a 4x4 box average
  - **BOX_5X5** - use a 5x5 box average
  - **BOX_6X6** - use a 6x6 box average
  - **BOX_7X7** - use a 7x7 box average

- **BOX_8X8** - use a 8x8 box average
- **BOX_9X9** - use a 9x9 box average
- **MAX_2X2** - (terrain layers only) use maximum value found in 2x2 box
- **MAX_3X3** - (terrain layers only) use maximum value found in 3x3 box
- **MAX_4X4** - (terrain layers only) use maximum value found in 4x4 box
- **MAX_5X5** - (terrain layers only) use maximum value found in 5x5 box
- **MAX_6X6** - (terrain layers only) use maximum value found in 6x6 box
- **MAX_7X7** - (terrain layers only) use maximum value found in 7x7 box
- **MAX_8X8** - (terrain layers only) use maximum value found in 8x8 box
- **MED_2X2** - (terrain layers only) use median value found in 2x2 box
- **MED_3X3** - (terrain layers only) use median value found in 3x3 box
- **MAX_9X9** - (terrain layers only) use maximum value found in 9x9 box
- **MED_4X4** - (terrain layers only) use median value found in 4x4 box
- **MED_5X5** - (terrain layers only) use median value found in 5x5 box
- **MED_6X6** - (terrain layers only) use median value found in 6x6 box
- **MED_7X7** - (terrain layers only) use median value found in 7x7 box
- **MED_8X8** - (terrain layers only) use median value found in 8x8 box
- **MED_9X9** - (terrain layers only) use median value found in 9x9 box
- **MIN_2X2** - (terrain layers only) use minimum value found in 2x2 box
- **MIN_3X3** - (terrain layers only) use minimum value found in 3x3 box
- **MIN_4X4** - (terrain layers only) use minimum value found in 4x4 box
- **MIN_5X5** - (terrain layers only) use minimum value found in 5x5 box
- **MIN_6X6** - (terrain layers only) use minimum value found in 6x6 box
- **MIN_7X7** - (terrain layers only) use minimum value found in 7x7 box
- **MIN_8X8** - (terrain layers only) use minimum value found in 8x8 box
- **MIN_9X9** - (terrain layers only) use minimum value found in 9x9 box

- ***ANTI_ALIAS*** [DEPRECATED - use SAMPLING_METHOD instead] (elevation and raster only) - specifies whether to remove jagged edges by making a subtle transition between pixels. Turning off this option helps maintain the hard edges of the pixels as they are rasterized. Use ANTI_ALIAS=YES to turn on. Anything else turns it off.

- **TRANSLUCENCY** (elevation and raster only) - specifies the level of translucency (i.e. how "see-through" the layer is). Value values range from *0 to 512*, with 0 meaning the layer is completely transparent (i.e. invisible) and 512 meaning the layer is completely opaque (this is the default).

- **IGNORE_ALPHA** (raster only) - specifies that an embedded alpha channel in an image should be ignored. This is useful for images that have incorrect alpha channels. Use IGNORE_ALPHA=**YES** to enable.

- **BLEND_MODE** (elevation and raster only)- specify blend mode to use for combining this overlay and any previously loaded overlays

  - **NO_BLEND** - no blending is done, this is the default
  - **MULTIPLY**
  - **SCREEN**
  - **OVERLAY**
  - **HARD_LIGHT**
  - **COLOR_BURN**
  - **COLOR_DODGE**
  - **DARKEN**
  - **LIGHTEN**
  - **DIFFERENCE**
  - **EXCLUSION**
  - **APPLY_COLOR**
  - **APPLY_COLOR_REVERSE**
  - **KEEP_RED**
  - **KEEP_GREEN**
  - **KEEP_BLUE**
  - **SPOT_NATURAL_COLOR_SPOT_TO_NATURAL**
  - **PSEUDO_NATURAL_COLOR_CIR_TO_NATURAL**
  - **COLOR_TO_GRAYSCALE**

- **FEATHER_BLEND_EDGES** (raster only) - specifies that the layer should be feature-blended around one or more ledges. This is a numeric bitfield value. Add (sum) the following values to enable blending on that edge:

  - **1** - blend top edge
  - **2** - blend bottom edge
  - **4** - blend left edge
  - **8** - blend right edge
  - **32** - just crop to feather boundary rather than feathering
  - **64** - feather outside polygon edge

For example, to blend all edges, use FEATHER_BLEND_EDGES=**15**. The FEATHER_BLEND_SIZE parameter is used to specify how many pixels to blend on each blended edge.

- **FEATHER_BLEND_SIZE** (raster only) - specifies the size in pixels to use for a blend boundary.

- **FEATHER_BLEND_POLY** (raster only) - specifies the name of a previously defined shape from "DEFINE_ SHAPE" on page 20 to feather too. You can also use FEATHER_BLEND_POLY=**COVERAGE** to calculate the polygonal coverage of the layer and feather blend to that automatically.

- **FEATHER_BLEND_POLY_FILE** (raster only) - specifies that the polygon boundary to feather blend this layer against should come from the specified file.

- **BAND_SETUP** (raster only) - specifies what bands of data from the raster file being loaded should be used to populate the red, green, and blue color channels when displaying the image. This is useful for multi-spectral imagery which may have more than 3 color bands. The default band setup will be to use the first three bands as follows: BAND_SETUP="**0,1,2**". Note that not all raster formats support specifying a non-default band setup.

- **CONTRAST_MODE** (raster only) - specifies the type of contrast adjustment to apply to the data.

    - **NONE** - no contrast adjustment applied (this is the default)
    - **PERCENTAGE** - apply a percentage contrast adjustment. The CONTRAST_STRETCH_SIZE parameter can be used to override the number of standard deviations from the mean to stretch to.
    - **MIN_MAX** - apply a min/max contrast stretch, stretching the available range of values in each color band to the full range of 0-255. For imagery which contains both black and white, this will have no affect.

- **CONTRAST_SHARED** (raster only) - specifies whether or not the contrast adjustment for this layer will share the adjustment with other contrast-adjusted layers in order to ensure a consistent modification across layers. Use CONTRAST_SHARED=**YES** to enable contrast sharing.

- **CONTRAST_STRETCH_SIZE** (raster only) - specifies the number of standard deviations from the mean to use in a PERCENTAGE contrast adjustment. The default is 2.0.

- *AUTO_CONTRAST* (raster only) - DEPRECATED, use CONTRAST_MODE instead - specifies whether to automatically calculate and apply a 2 standard deviation contrast adjustment to the image. Use AUTO_ CONTRAST=YES to turn on. Anything else turns it off.

- *COLOR_INTENSITY*(DEPRECATED use COLOR_INTENSITY_FULL parameter) (elevation and raster only). - specifies the color intensity to use when adjusting the brightness of pixels in the overlay. Valid values range from **0 to 20**, with 0 being completely black, 10 being no alteration, and 20 being completely white. For example, to make an image slightly darker, you could use COLOR_INTENSITY=7.

- **COLOR_INTENSITY_FULL** (elevation and raster only) - specifies the color intensity to use when adjusting the brightness of pixels in the overlay. Valid values range from **0 to 512**, with 0 being completely white, 256 being no alteration, and 512 being completely black. For example, to make an image slightly darker, you could use COLOR_INTENSITY=300. (NOTE: This parameter replaces the COLOR_INTENSITY parameter).

- **TEXTURE_MAP** (raster only) - specifies that this image should be draped over any elevation data loaded before it. Use TEXTURE_MAP=**YES** to turn on. Anything else turns it off.

- **PALETTE_NAME** (palette-based raster only) - specifies the filename of a recognized palette file to override the default colors in this layers palette or a palette previously defined with the DEFINE_PALETTE command. Use this to change the color interpretation of palette indices.

- **TRANSPARENT_COLOR** (elevation and raster only) - specifies the color to make transparent when rendering this overlay. The color should be specified as *RGB(<red>,<green>,<blue>)*. For example, to make white the transparent color, use TRANSPARENT_COLOR=RGB(255,255,255). If you do not wish any color to be transparent, do not use this parameter. Optionally, if the image that you are making transparent uses a palette for the colors, you can specify a palette index in the following format: *INDEX(<0-based palette index>)*. For example, to make the second color in the palette transparent, use TRANSPARENT_COLOR=INDEX (1).

- **TRANSPARENT_COLOR_DIST** - for layers that have specified a color to make transparent, this parameter allows you to specify how far a color in the layer has to be from the specified TRANSPARENT_COLOR value to be treated as transparent as well. The default value of **0** means that the colors have to exactly match for the pixel to be treated as transparent. Larger values (*up to* **256**) allow larger distances between the layer color and the transparent color. This is useful for lossy formats, like JPEG.

- **COLOR_GRADE** (raster only) - specifies the color grading values to use for this layer (as configured on the Color Grade options dialog tab). This should be a comma-delimited list with the saturation value (from

0-**1**) first, then the input and output range for the red, green, and blue color channels, as follows: COLOR_ GRADE=*saturation,red_in_start,red_in_end,red_out_start,red_out_end,...,blue_out_end*

- **CLIP_COLLAR** (raster only) - specifies whether to clip the collar off of the image. The following values are supported for cropping:
    - **NONE** - no collar cropping is performed.
    - **AUTO** - automatically remove a USGS DRG-style collar or a 3.75 DOQQ collar
    - **LAT_LON** - crop the collar to a a specified set of bounds specified in arc degrees in the native datum of the layer. The bounds should be specified using the CLIP_COLLAR_BOUNDS parameter.
    - **NATIVE** - crop the collar to a specified set of bounds specified in the native projection system and datum of the layer. The bounds should be specified using the CLIP_COLLAR_BOUNDS parameter.
    - **PIXELS** - crop a given number of pixels off of each side of the layer. The number of pixels to remove from each side should be specified using the CLIP_COLLAR_BOUNDS parameter.
    - **SNAP_DEGREES** - crop the collar by snapping each edge to a specified degree boundary specified in arc degrees in the native datum of the layer. The bounds should be specified using the CLIP_COLLAR_BOUNDS parameter. For example to crop the west and east edges to a half degree boundary and the north and south edges to a one degree boundary, use the following: CLIP_COLLAR_BOUNDS=0.5,1.0,0.5,1.0.
    - **POLY** - crop to a polygon provided with the CLIP_COLLAR_POLY parameter.
- **CLIP_COLLAR_BOUNDS** (raster only) - specifies the bounds of the collar to be clipped off when the CLIP_COLLAR parameter is enabled. The coordinates should be specified in arc degrees, native layer coordinates, or pixel counts as a comma-delimited list going *west,south,east,north*. For example, CLIP_ COLLAR_BOUNDS=34.25,-109.0,34.375,-108.875.
- **CLIP_COLLAR_POLY** (raster only) - specifies the name of the previously defined shape (with the DEFINE_SHAPE command) to crop the layer to when the CLIP_COLLAR=POLY parameter is used. The coordinates in the shape must have been defined in the native projection system of the layer being loaded. Unless you provide CLIP_COLLAR_POLY_SIMPLIFY=NO, the clip polygon will be simplified to 1/10th of a pixel resolution to reduce the size of the crop polygon for faster cropping without noticeably changing the shape.
- **CLIP_COLLAR_POLY_EXCLUDE** (raster only) - specifies that the crop to the polygon specified with CLIP_COLLAR_POLY should keep all parts of the layer outside the crop polygon(s) rather than what is inside the polygon(s). Add CLIP_COLLAR_POLY_EXCLUDE=**YES** to enable this behavior.
- **CLIP_COLLAR_POLY_SIMPLIFY** - specifies that the clip polygon will be simplified to 1/10th of a pixel resolution to reduce the size of the crop polygon for faster cropping without noticeably changing the shape. Enabled by default, use CLIP_COLLAR_POLY_SIMPLIFY=**NO** to disable.
- **CLIP_COLLAR_POLY_PIXEL** (raster only) - specifies that the coordinates in the crop polygon from the CLIP_COLLAR_POLY parameter are in pixel relative coordinates for the layer rather than in the native system of the layer. Use this if you need to crop a layer to a particular boundary in known pixel coordinates. The coordinates will convert to native layer coordinates on load.

Vector Parameters

- **VIDEO_FILENAME** - specifies the full path and filename or URL for a video file to associate with the layer. This video can then be displayed for selected point or line features with a video timestamp.

Vector Label Parameters

The parameters below allow specifying how to create display labels for vector layers.

- **LABEL_FIELD** - specifies the *name of the attribute field* to use as the label attribute for the features in the file. By default the attribute-based labeling will only be applied to those features that don't already have a label, but if the LABEL_FIELD_FORCE_OVERWRITE attribute is set to YES then all features will have their labels replaced. If you want to build the label from multiple attributes, separate them with '>+<' in the file, like LABEL_FIELD='RD_PREFIX>+<RD_NAME>+<RD_SUFFIX'.
- **LABEL_CUSTOM_DEF** - specifies a custom free-form string describing how to form the display labels for this layer. This can include embedded attribute values as *%ATTR_NAME%*.
- **LABEL_FIELD_FORCE_OVERWRITE** - specifies that the LABEL_FIELD or LABEL_CUSTOM_DEF attribute value should be applied to all feature labels, not just those that don't already have labels. Use LABEL_ FIELD_FORCE_OVERWRITE=**YES** to enable.

- **SHOW_LABELS** - specifies whether or not labels are shown for features in this layer, assuming they would be otherwise shown. The default is SHOW_LABELS=**YES**. Use SHOW_LABELS=**NO** to disable the display of labels for this layer regardless of other settings.

- **LABEL_PREFIX** - specifies the prefix to prepend to attribute-based labels

- **LABEL_SUFFIX** - specifies the suffix to append to attribute-based labels

- **LABEL_PRECISION** - value is an integer indicating the number of decimal digits to use. This applies to numeric labels.

- **LABEL_REMOVE_TRAILING_ZEROS** - This removes the trailing zeros to the right of the decimal place in numeric labels. This can be specified by listing the parameter alone, or accepts boolean values.

- **LABEL_USE_SCIENTIFIC_NOTATION** - Display the number in scientific notation. This accepts boolean values, or can be called by listing the parameter alone.

Lidar Display Parameters

The parameters below allow specifying options for working with Lidar data.

- **LIDAR_DRAW_MODE** - specifies how points in a Lidar point cloud layer should be drawn. The following values are supported:
    - **COLOR** - if the points have an associated RGB color, use that. Otherwise color by elevation.
    - **ELEV** - color by elevation of the point using the current elevation shader.
    - **INTENSITY** - color as a grayscale image by the intensity
    - **CLASS** - color by the point classification
    - **RETURN** - color by the return number
    - **HEIGHT_ABOVE_GROUND** - color by the height above ground
    - **POINT_SOURCE_ID** - color by the point source ID
    - **POINT_INDEX** - color by the index of the point in the cloud
    - **RETURN_HEIGHT_DELTA** - color by the difference in height between first and last return
    - **CIR** - color as color infrared if NIR band present
    - **NDVI** - color by calculated NDVI value if NIR band present
    - **NDWI** - color by calculated NDWI value if NIR band present
    - **DENSITY** - color by point density
    - **WITHHELD** - color by withheld flag
    - **OVERLAP** - color by overlap flag
    - **KEY_POINT** - color by key point flag

- **LIDAR_POINT_SIZE** - specifies how large the points in a Lidar point cloud are drawn. The default value of **0** will automatically scale the points to be larger as you zoom in on them. Specify a fixed number to always draw them at a particular size in pixels.

- **LIDAR_DRAW_QUALITY** - specifies the quality setting (**0-100**) for drawing the Lidar point cloud. Larger values draw a larger fraction of the points when zoomed out, but the draw will take longer to complete.

- **LIDAR_FILTER** - specifies a comma-separated list of Lidar class numbers to enable or disable for load. Provide a minus sign (**-**) to remove the type from the filter rather than add it. The filter starts off with nothing in it if you provide a LIDAR_FILTER string, but you can add **ALL** to enable everything or **NONE** to clear the filter, then add or remove stuff after that. For example, to specify a class filter with only types 2 and 3 enabled, use `LIDAR_FILTER="NONE,2,3"`. To get one with everything but classes 2 and 3, use `LIDAR_FILTER="ALL,-2,-3"`.

- **LIDAR_RETURN_FILTER** - specifies a comma-separated list of Lidar return types to enable or disable for load. Provide a minus sign (**-**) to remove the type from the filter rather than add it. The filter starts off with loading everything, but you can add **ALL** to enable everything or **NONE** to clear the filter, then add or remove stuff after that. For example, to specify a return filter with only unknown and first returns, use `LIDAR_RETURN_FILTER="NONE,0,1"`. To get one with everything but the first return, use `LIDAR_RETURN_FILTER="ALL,-1"`. The numeric values have the following meanings:
    - **0** - Unknown Returns
    - **1** - First Return
    - **2** - Second Return
    - **3** - Last Return
    - **4** - Single Return
    - **5** - First of Many Returns
    - **6** - Second of Many Returns
    - **7** - Third of Many Returns
    - **8** - Last of Many Returns

- **LIDAR_USE_INTEN_FOR_NIR** - specifies that we should treat the intensity value Lidar points as the NIR (near infrared) value if the point cloud has RGB colors but no NIR values of its own. Use LIDAR_USE_INTEN_FOR_NIR=**YES** to enable.

Layer Rectification/ Control Points

The parameters below allow defining a series of control points and rectification parameters for setting up a coordinate mapping from pixel space to real-world projection coordinates for the layer.

- **GCP** - specifies a single ground control point for use in rectifying a file. The GCP record consists of 5 comma-delimited fields, *the control point name, the pixel X and Y coordinates, and the corresponding ground X and Y coordinates*. A separate GCP parameter and value should be used for each control point used in the rectification. As an alternative, the GCP_FILENAME parameter (see below) can be used instead.

- **GCP_FILENAME** - specifies the name of a control point file used to rectify the file being imported.

- **GCP_PROJ_NAME** - specifies the name of the projection that the ground control points are provided in. This name must have been defined with a prior DEFINE_PROJ command. Use this if you want to specify control points in a projection other than what you want to define as the native projection for the file. Note that you must also explicitly specify the name projection of the file using either the PROJ, PROJ_NAME, PROJ_EPSG_CODE or PROJ_FILENAME parameters.

- **TRANSFORM_FILENAME** - specifies the name of a control point file used to transform the coordinates of the imported file. This is different than the GCP_FILENAME in that the file defines a mapping of world coordinates to a new set of world coordinates rather than pixel coordinates to world coordinates. Each line should be of the format: *x_orig,y_orig,x_new_y_new*

- **GCP_PROJ_FILENAME** - specifies the name of the projection (.prj) file that contains the projection definition for the projection that the ground control points are provided in. Use this if you want to specify control points in a projection other than what you want to define as the native projection for the file. Note that you must also explicitly specify the name projection of the file using either the PROJ, PROJ_NAME, PROJ_EPSG_CODE or PROJ_FILENAME parameters.

- **GCP_PROJ_EPSG_CODE** - specifies the EPSG code of the projection that the ground control points are provided in. Use this if you want to specify control points in a projection other than what you want to define as the native projection for the file. Note that you must also explicitly specify the name projection of the file using either the PROJ, PROJ_NAME, PROJ_EPSG_CODE or PROJ_FILENAME parameters.

- **RECTIFY** - specifies the rectification method to use for rectifying this file. Valid value are **LINEAR**, **HELMERT**, **AFFINE**, **POLYNOMIAL**, and **TRIANGULATION**. If you do not specify a rectification type but do provide at least two ground control points, the best rectification method will automatically be chosen based on the number of control points specified.

- **RECTIFY_4_POINT_POLY_ONLY** - specifies that if RECTIFY=POLYNOMIAL is used to specify the rectification method, the polynomial will always be a 1st degree polynomial and won't switch automatically to a 2nd degree polynomial at 6 or more points. By default, the 2nd degree polynomial will automatically be used

SHIFT_LAYER

The SHIFT_LAYER command moves a layer by the specified offset.

The following parameters are supported by the command:

- **FILENAME** - filename of the layer to split. If an empty value is passed in, all loaded vector layers will be split. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer split or '**SELECTED LAYERS**' to have any layers selected in the Control Center split. If you don't pass anything in all loaded layers will be operated on.

- **COORD_OFFSET** - specifies the offset to apply to any coordinates for the features that match the specified criteria. The offset should be specified as a *comma-delimited list of the X and Y offsets*, such as `COORD_OFFSET="100000.0,200000.0"`.

- **COORD_OFFSET_UNITS** - String representing units to use to interpret value obtained by "COORD_OFFSET". Accepts "**M**" (meters), "**METERS**", "**FT**", "**FEET**", "**KM**", "**MILES**", "**NM**" (nautical miles), "**DEGREES**" (arc degrees) or any other unit name from the projection dialog. Defaults to meters. e.g. `COORD_OFFSET_UNITS="FT"`

Example: Shift a layer by 500 meters in the X direction and 300 meters in the Y direction.

```
SHIFT_LAYER FILENAME="BackCove_base.tif" COORD_OFFSET="500,300" COORD_OFFSET_UNITS="M"
```

QUERY_LAYER_METADATA

The QUERY_LAYER_METDATA command allows a layer metadata value to be stored in a script variable. The user needs to identify the layer based on its file name (or layer description) and the metadata attribute based on the name it has when displayed via the Control Center. The following parameters are supported by this command:

- **RESULT_VAR** - specifies the name of the script variable where the metadata will be stored.
- **METADATA_LAYER** - identifies the layer from which the metadata will be copied. This is the same value that would be passed to the FILENAME parameter on the IMPORT command used to load the layer.
- **METADATA_ATTR** - This is the string used to identify the metadata from which the value will be copied. The complete list of metadata attribute names for a layer can be seen by clicking the Metadata... button on the Control Center, or by running the EXPORT_METADATA command and looking at the result file. Some example metadata attributes are "**UPPER LEFT X**", "**AREA COUNT**", "**PROJ_DESC**", etc.

SAMPLE

Example: Stores the layer's DESCRIPTION metadata in a variable called %DESC%:

```
DEFINE_VAR NAME="LAYER" VALUE="P:\Data\Areas.shp"
IMPORT FILENAME="%LAYER%" TYPE="SHAPEFILE"
QUERY_LAYER_METADATA METADATA_LAYER="%LAYER%" METADATA_ATTR="DESCRIPTION" RESULT_VAR="DESC"
```

UNLOAD_ALL

The UNLOAD_ALL command unloads all currently loaded data. This command may be called with no parameters.

The following parameters are supported by the command:

- **VECTOR_ONLY** - specifies that only layers containing vector data shall be unloaded. All raster and gridded elevation layers will remain loaded. Use VECTOR_ONLY=**YES** to enable unloading just the vector layers.

UNLOAD_LAYER

The UNLOAD_LAYER command allows you to unload all previous loaded layers with a given filename. This is useful if you don't want to unload all previously loaded layers just to get rid of a few of them.

The following parameters are supported by the command:

- **FILENAME** - filename of the layer to unload. If an empty value is passed in, all layers that were created by the script, such as those from a GENERATE_CONTOURS command, will be unloaded. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center unloaded. You can also pass in the full description of the loaded layer to use in case you want to unload a layer not loaded from a file.
- **SHOW_WARNINGS** - specifies whether or not to show a warning if no layers matching the FILENAME are found. Defaults to on, use SHOW_WARNINGS=**NO** to disable.

SPLIT_LAYER

The SPLIT_LAYER command allows you to split a layer into multiple layers based on some attribute value. The original layer is closed if there was any split that happened.

The following parameters are supported by the command:

- **FILENAME** - filename of the layer to split. If an empty value is passed in, all loaded vector layers will be split. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer split or '**SELECTED LAYERS**' to have any layers selected in the Control Center split. If you don't pass anything in all vector layers will be operated on.

- **SPLIT_BY_ATTR** - specifies the attribute to split on. A new layer will be generated for each set of attributes values in the input data. This can be a normal attribute name or a special one. See special [Attribute Name](#) parameter details.

- **SPLIT_ATTR_SEP** - specifies a separator to check for in the split attribute. If multiple values are in the split attribute a separate layer will be created for each. You can use special values like **COMMA**, **TAB**, **SPACE**, and **SEMICOLON** in addition to just *specifying the separator string* directly. For example if you use SPLIT_BY_ATTR="PROPERTY_ID" and a feature has a PROPERTY_ID attribute with the value "A,B,C" and you add `SPLIT_ATTR_SEP=COMMA`, you will get 3 copies of the feature, one in layer A, one in layer B, and one in layer C.

- **LAYER_DESC_ATTR_ONLY** - specifies that the description assigned to newly created layers will just consist of the attribute value rather than the original layer description with the attribute value appended.

- **CLOSE_ORIG_LAYER** - specifies that the layer that was split into new sub-layers should be closed when the operation completes. This is done by default, so add CLOSE_ORIG_LAYER=**NO** to keep the original layer open.

SORT_LAYERS

The SORT_LAYERS command allows you to sort the loaded layers based on some criteria, like name, resolution, type, etc. The following parameters are supported by the command:

- **FILENAME** - filename of the layers to sort. If you leave this off all layers will be sorted. You can use wildcards to match on multiple loaded files. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`.

- **LAYER_GROUP** - specifies the layer group to match on. Only loaded layers in the specified group will be matched on. You can use wildcards (like '*' or '?') to match multiple groups. If you don't provide this parameter any group is ok. If you want only those layers not in a group, add `LAYER_GROUP=""` for none.

- **MAP_ORDER** - specifies how to sort the maps. The following values are recognized:
    - **LAYER_DESC_ASC** - ascending order by layer description
    - **LAYER_DESC_DESC** - descending order by layer description
    - **RESOLUTION** - sort in order of increasing resolution (i.e. less detailed drawn first)
    - **TYPE** - sort by type (elevation first, then raster, then vector)
    - **MOVE_FRONT** - move the matching layers to the front of the draw order
    - **MOVE_FRONT_GROUP** - move the matching layers to the front of the group specified with LAYER_GROUP
    - **MOVE_END** - move the matching layers to the end of the draw order
    - **MOVE_END_GROUP** - move the matching layers to the end of the group specified with LAYER_GROUP
    - **NSWE** - sort by map box, north to south, west to east in row
    - **NSEW** - see NSWE, just different order
    - **SNWE**
    - **SNEW**
    - **WENS**
    - **WESN**
    - **EWNS**
    - **EWSN**

EDIT_MAP_CATALOG

The EDIT_MAP_CATALOG command allows you to work with map catalogs (managed collections of map files), including create new map catalogs, adding maps to existing map catalogs, and removing maps from existing map catalogs.

The following parameters are supported by the command:

- **FILENAME** - filename of the map catalog to create/update.

- **CREATE_IF_EMPTY** - specifies whether or not the map catalog (.gmc) file should be created if it doesn't already exist. The default is **YES**. If you specify the FILENAME of a map catalog file that doesn't exist and have CREATE_IS_EMPTY=**NO** provided, nothing will be done and an error will be logged.

- **ZOOM_DISPLAY** - specifies when the maps in the map catalog should be displayed and when it should be hidden based on the display zoom scale. This command will be formatted as a name from the list, below followed by 2 numeric parameters. For example, use `ZOOM_DISPLAY="SCALE,25000,0"` to have a map display only when zoomed in below 1:25000 scale.

    - **ALWAYS** - always display the map. The numeric parameters are ignored.

    - **PERCENT** - display the map when the map bounding box is a certain percentage of the screen size. For example, use `ZOOM_DISPLAY="PERCENT,0.10,0"` to display the map when its bounding box is at least 10% of the screen size.

    - **PIXEL_SIZE** - display the map when each display pixel is less than some number of meters in size. For example, use `PIXEL_SIZE="SCALE,10,0"` to display the map when the current display resolution is 10 meters per pixel (or less/higher resolution).

    - **SCALE** - display the map when the current display is at or below a certain scale. For example, use `ZOOM_DISPLAY="SCALE,25000,0"` to display the map when the current draw scale is at or below 1:25000.

    - **SCALE_RANGE** - display the map when the current display is below a range of scale value. For example, use `ZOOM_DISPLAY="SCALE_RANGE,25000,100000"` to display the map when the current draw scale is between 1:25000 and 1:100000.

- **ADD_FILE** - specifies the full path to a file to add to the map catalog. You can include wildcard characters, like * and *?*, in the filename. You can also *include multiple* ADD_FILE parameters to add multiple files in one command. For example, to add all of the ECW files in a folder to the catalog, use `ADD_FILE-E="C:\path_to_files\*.ecw"`.

- **ADD_FILE_LIST** - specifies the full path to a text file listing maps to add to the map catalog. The text file should contain the full path to each file to add on each line of the file.

- **REMOVE_MAP** - specifies the full path of a file to remove from the map catalog. You can include wildcard characters, like * and *?*, in the filename. You can also include multiple ADD_FILE parameters to remove multiple files in one command. For example, to remove all of the ECW files in the map catalog, use `REMOVE_MAP="*.ECW"`. To clear out a map catalog, use `REMOVE_MAP="*"`. To remove a file by filename without specifying a folder, use a * for the path. For example to remove "c:\temp\my_map.tif", you could use `REMOVE_MAP="*\my_map.tif"` or `REMOVE_MAP="c:\temp\my_map.tif"`.

- **LOAD_FLAGS** - specifies the load flags to use for the file. These are the same as used for the IMPORT command. Use this to skip any load prompts. The values are format specific.

SAMPLES

Here is a sample showing how to create a map catalog and then load it:

```
// Create the map catalog. Maps should show when they take up at least 10% of display.
EDIT_MAP_CATALOG FILENAME="C:\TEMP\EXPORT TEST\script_catalog.gmc" CREATE_IF_EMPTY=YES \
 ADD_FILE="c:\temp\export test\*.tif" ADD_FILE="c:\temp\export test\*.dem" \
 ZOOM_DISPLAY="PERCENT,0.10,0"
// Load the map catalog
IMPORT FILENAME="c:\temp\export test\script_catalog.gmc"
```

Terrain and 3D Analysis

CALC_VOLUMES

The CALC_VOLUMES command allows you to calculate the volume for each area in the specified layer using the currently loaded terrain data.

The following parameters are supported by the command:

- **FILENAME** - filename of the layer containing the areas to be used in the calculations. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`.
- **OUTPUT_FILENAME** - specifies the path and file name of the output file for volume statistics. At least one of this and the ADD_VOLUME_ATTRS parameter is required. Using both is valid also.
- **ADD_VOLUME_ATTRS** - indicates whether or not to add the volume measurements to the area features as attributes. Use ADD_VOLUME_ATTRS=**YES** to add the volume data to the feature attribute list. At least one of this and the OUTPUT_FILENAME parameter is required. Using both is valid also.
- **BASE_ELEVATION** - specifies the base elevation in meters for volume calculations. This parameter is optional. The default is **0**.
- **VOLUME_UNIT** - specifies the unit to be used for the volume calculations. This parameter is optional, and the default is **CUBIC_METERS**. Valid values are:
  - **ACRE_FEET**
  - **ACRE_INCHES**
  - **BARRELS**
  - **BARRELS_OIL**
  - **CUBIC_FEET**
  - **CUBIC_METERS**
  - **CUBIC_YARDS**
  - **GALLONS**

SAMPLE

Here is a sample of calculating volumes in cubic feet, adding the results to the features as attributes.

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
CALC_VOLUMES FILENAME="boundaries 2.dxf" ADD_VOLUME_ATTRS=YES
```

CALC_VOLUME_BETWEEN_SURFACES

Calculates the volume between two elevation grids. If you specify an area layer, the volume will be calculated for each feature, and volume attributes will be added to the feature. The following parameters are supported by the command:

- **LAYER1_FILENAME** - specifies the base elevation grid. Elevations in LAYER2_FILENAME will will be subtracted from elevations in this layer during the volume calculations.
- **LAYER2_FILENAME** - specifies the elevation grid with values that will be subtracted from LAYER1_ FILENAME.
- **AREA_FILENAME** - An optional layer that contains area features. If it is specified, the volume will be calculated for each area, and the volume attributes will be added to the feature.
- **VOLUME_UNIT** - specifies the unit to be used for the volume calculations. This parameter is optional, and the default is **CUBIC_METERS**. Valid values are:
  - **ACRE_FEET**
  - **ACRE_INCHES**
  - **BARRELS**
  - **BARRELS_OIL**
  - **CUBIC_FEET**
  - **CUBIC_METERS**
  - **CUBIC_YARDS**
  - **GALLONS**
- **OUTPUT_FILENAME** - specifies the path and file name of the output file for volume statistics.

Specify Bounds for Operation

See also "Specify Bounds for Operation" on page 125

SAMPLE

```
CALC_VOLUME_BETWEEN_SURFACES LAYER1_FILENAME="P:\Data\baseGrid.tif" \
LAYER2_FILENAME="P:\Data\lidarGrid2.dem" \
AREA_FILENAME="P:\Data\ClipAreas.shp" \
VOLUME_UNIT="CUBIC_FEET" \
OUTPUT_FILENAME="%OUTDIR%\fromScript_wClip.csv"
```

COMBINE_TERRAIN

The COMBINE_TERRAIN command generates a new terrain (gridded elevation) layer by combining two loaded terrain layers through some operation, like addition, subtraction (difference), average, min/max, etc. The new terrain layer can then be operated on just like any other terrain layer.

The following parameters are used by the COMBINE_TERRAIN command:

- **LAYER1_FILENAME** - full path and filename of the first loaded terrain layer to use. You can also pass in the full description of the loaded layer to use in case you want to use a layer not loaded from a file. If you are using one of the combine operations that works on multiple layers in a single list, like ADD, AVERAGE, MINIMUM, MAXIMUM, or COUNT_VALID, you can leave this blank and have all loaded layers examined.
- **LAYER2_FILENAME** - full path and filename of the second loaded terrain layer to use. You can also pass in the full description of the loaded layer to use in case you want to use a layer not loaded from a file. You do not have to provide this for those operations that work on multiple layers in a single list, like ADD, AVERAGE, MINIMUM, MAXIMUM, or COUNT_VALID.
- **COMBINE_OP** - defines the operation to perform when combining the layers. The following operations are supported:
  - **ADD** - adds the values from the first layer to the second
  - **SUBTRACT_SIGNED** - subtracts the values of the second layer from the first and saves the signed result.
  - **SUBTRACT_UNSIGNED** - subtracts the values of the second layer from the first and saves the magnitude of the result.
  - **AVERAGE** - saves the average of the values from the first and second layers.
  - **MINIMUM** - saves the smaller of the values from the first and second layers.

- **MAXIMUM** - saves the larger of the values from the first and second layers.
- **MULTIPLY** - multiplies the values from the first and second layers. If one or both of the values is missing, the sample is marked as invalid.
- **DIVIDE** - divides the value from the first layer by the value in the second layer. If one or both of the values is missing or if the second value is 0, the sample is marked as invalid.
- **FILTER_KEEP_FIRST** - saves the first layer value if the second layer value is valid.
- **FILTER_KEEP_FIRST_IF_SECOND_INVALID** - saves the first layer value if the second layer value is invalid
- **FILTER_KEEP_FIRST_IF_GT_SECOND** - saves the first layer value if the second layer value is valid and the first layer value is greater than the second layer value.
- **FILTER_KEEP_FIRST_IF_LT_SECOND** - saves the first layer value if the second layer value is valid and the first layer value is less than the second layer value.
- **FILTER_KEEP_FIRST_IF_SECOND_GT_VAL** - saves the first layer value if the second layer value is valid and the second layer value is greater than the value provided with the COMPARE_VAL parameter
- **FILTER_KEEP_FIRST_IF_SECOND_LT_VAL** - saves the first layer value if the second layer value is valid and the second layer value is less than the value provided with the COMPARE_VAL parameter
- **COUNT_VALID** - counts the number of layers that have a valid sample at each grid location. This works with both raster and elevation data.

- **COMPARE_VAL** - provides a numeric vlaue to compare against for some of the combine operations above.

- **LAYER_DESC** - specifies the name to assign to the newly generated terrain layer. If no layer description is provided, the default name of "**Combined Elevation Grid**" will be used.

- **ELEV_UNITS** - specify elevation units to use in new terrain layer
  - **FEET** - export in US feet
  - **DECIFEET** - export in 10ths of US feet
  - **METERS** - export in meters
  - **DECIMETERS** - export in 10ths of meters
  - **CENTIMETERS** - export in centimeters

- **SPATIAL_RES** - specifies spatial resolution. Defaults to the minimum spatial resolution of the two layers if not specified. Should be formatted as *x_resolution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to export at 30 meter spacing, the parameter/value pair would look `like SPATIAL_RES=30.0,30.0`. You can also specify as a *percentage* of the default resolution by adding a percent. For example to get half the detail your double the spatial resolution value, so you would use `SPATIAL_RES="200%,200%"`.

- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will automatically be converted to the current view/export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.

- **FILL_GAPS** - specifies that small gaps in between and within the data sets being combined will be filled in by interpolating the surrounding data to come up with an elevation for the point in question. This option is off by default, specify FILL_GAPS=**NO** to turn off.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

GENERATE_CONTOURS

The GENERATE_CONTOURS command allows for the generation of contour lines (isolines of equal elevation) from any or all currently loaded elevation data. The following parameters are supported by the command.

- **FILENAME** - filename of the layer(s) to contour. If an empty value is passed in, all loaded terrain data will be used. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`.

- **ELEV_UNITS** - specify elevation units to use in export
  - **FEET** - export in US feet
  - **METERS** - export in meters

- **INTERVAL** - specifies the contour interval to use. This must be a *whole number greater than 0*. The units are specified with the ELEV_UNITS parameter described above. If you wanted to generate a contour file with an interval of 20 feet, you would use `INTERVAL=20 ELEV_UNITS=FEET` in the parameter list. If no interval is provided, a default one is guessed based on the elevation range of the loaded elevation data.

- **GEN_FROM_TIN_AREAS** - specifies that the contours should be generated from any loaded TIN (3D triangle) areas rather than from loaded terrain data. Add GEN_FROM_TIN_AREAS=**YES** to command to use TINs.

- **MULT_MINOR** - specifies how many contours apart every intermediate contour is. For example, use `MULT_MINOR=5` to make every 5th contour an intermediate contour. So using `INTERVAL=10` and `MULT_MINOR=5` creates intermediate contours every 50 meters.

- **MULT_MAJOR** - specifies how many contours apart every major contour is. For example, use `MULT_MAJOR=10` to make every 10th contour an intermediate contour. So using `INTERVAL=10` and `MULT_MINOR=10` creates major contours every 100 meters.

- **MIN_ELEV** - minimum elevation to consider for contours. Must be specified in conjunction with MAX_ELEV in order to restrict the range of contour generation to anything other than the full range of loaded elevation values. Units are specified by ELEV_UNITS parameter.

- **MAX_ELEV** - maximum elevation to consider for contours. Must be specified in conjunction with MIN_ELEV in order to restrict the range of contour generation to anything other than the full range of loaded elevation values. Units are specified by ELEV_UNITS parameter.

- **SPATIAL_RES** - specifies spacing of grid points used to determine contour position. A smaller grid spacing results in higher fidelity, but larger, contours. Typically you'll want to use the default value which is the minimum spatial resolution of all loaded data. Should be formatted as *x_resolution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to use a grid with a 30 meter spacing, the parameter/value pair would look like `SPATIAL_RES=30.0,30.0`. You can also specify as a percentage of the default resolution by adding a *percentage*. For example to get half the detail, double the spatial resolution value, so you would use `SPATIAL_RES="200%,200%"`.

- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will automatically be converted to the current view/export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.

- **SIMPLIFICATION** - specifies the simplification threshold to use when generating the contours. This specifies how far off a straight line (in the units of the current projection) that a point has to be before it is kept. Generally you should not specify a simplification value as the default value of one tenth of the sample spacing works quite well. This is an option for advanced users only.

- **SAMPLING_METHOD** (elevation and raster only) - specifies the sampling method to use when resampling this layer. The following values are supported:
  - **DEFAULT** - Use either automatic resampling based on export or layer resampling, depending on setting of global flag about whether to resample on export
  - **AUTO** - Automatically select a resampling method based on how the export resolution and bounds compare to the original layout for a layer. For example if you export to a lower resolution a box averager of appropriate size may be used automatically
  - **LAYER** - Use the sampling method set for each layer
  - The list of SAMPLING_METHOD values for the IMPORT command can also be specified to use a particular sampling method for all layers being exported.
  - **Shared IMPORT SAMPLING_METHOD values**

    See "SAMPLING_METHOD (elevation and raster only) - specifies the sampling method to use when resampling this layer. " on page 39

- **GEN_HEIGHT_AREAS** - generate area features colored based on the current elevation shader in addition to generating contour lines. Use a value of **YES** to enable the generate of the height areas.

- **GEN_SPOT_ELEVATIONS** - generate spot elevations at min/max elevations. Use a value of **YES** to enable the generate of min/max spot elevation points.

- **SINGLE_LEVEL_ONLY** - specifies that the INTERVAL value is actually a value indicating the only height that a contour should be generated at. Use a value of **YES** to turn this functionality on.

- **FILL_GAPS** - specifies that small gaps in between and within the data sets being used to generate the contours will be filled in by interpolating the surrounding data to come up with an elevation for the point in question. This option is on by default, specify FILL_GAPS=**NO** to turn off.

- **LAYER_DESC** - specifies the name to assign to this layer. If no layer description is provided, the default name of "**GENERATED CONTOURS**" will be used.

- **INC_UNIT_SUFFIX** - specifies whether or not a unit suffix (either "m" or "ft") should be appended to the numeric label of generated features. By default this is enabled, so specify INC_UNIT_SUFFIX=**NO** to turn

unit suffixes off. This is useful if the data the contours are being generated over doesn't actually represent elevation.

- **SMOOTH_CONTOURS** - specifies whether or not generated contour line and area features should have smoothing applied to improve appearance. This option is enabled by default. Use SMOOTH_ CONTOURS=**NO** to disable smoothing.

- **CREATE_ON_WAY_DOWN** - specifies whether contours are created as the terrain passed from a higher elevation to a contour height (CREATE_ON_WAY_DOWN=**YES**) or the default way of being created when the terrain passes from a contour height to lower elevation values (CREATE_ON_WAY_DOWN=**NO**).

- **MIN_CONTOUR_LEN** - specifies that any closed contour lines less than the specified length (in meters) will be marked as deleted. This is useful for cleaning up contours in very rugged and detailed terrain, like from Lidar. The default is to keep all generated contour lines.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

GENERATE_WATERSHED and GENERATE_RIDGE_LINES

The GENERATE_WATERSHED command allows for the generation of a watershed, including stream flow and option-ally watershed areas for each stream segment. The GENERATE_RIDGE_LINES command shares the same para-meters, but finds ridge lines rather than stream lines. A ridge line is created wherever flow would accumulate in the inverse of the terrain surface. The following parameters are supported by the command.

- **FILENAME** - filename of the layer(s) to process. If an empty value is passed in, all loaded terrain data will be used. This parameter can be listed more than once to specify multiple input files, like `FILENAME`= `E="FILENAME_1" FILENAME="FILENAME_2"`.

- **STREAM_THRESHOLD** - specifies the number of cells that have to drain to a particular location before that location is considered to be part of the stream/ridge network.

- **MAX_DEPTH** - specifies the maximum depression depth (in meters) that will be filled prior to calculating the watershed. This is needed to prepare a terrain surface for flow analysis so that a continuous flow can be found. Note that while using a large MAX_DEPTH value may generate better results, it could also cause the process to take a lot longer.

- **GEN_AREAS** - generate watershed area features for each found stream segment outlining the area that drains into that stream. Enabled by default for watersheds and disabled for ridge lines, use GEN_ AREAS=**NO** to disable.

- **SMOOTH_STREAMS** - specifies whether or not generated streams/ridge lines should have smoothing applied to improve appearance. This option is enabled by default. Use SMOOTH_STREAMS=**NO** to disable smoothing.

- **FLOW_TO_POS** - specifies the X and Y location (or longitude/latitude) of a position to generate an area with all parts of the terrain surface that flow to (or near) that point in the vector area feature. The format is FLOW_TO_POS="*x,y*". You can provide multiple FLOW_TO_POS parameters to find the flow to multiple loc-ations. Use FLOW_TO_POS_THRESH to specify how close to the position the flow has to go in order to con-sider the source point part of that area. Use FLOW_TO_POS_PROJ to specify the projection of the position. If that is not provided it will be assumed to be in the current projection.

- **FLOW_TO_POS_THRESH** - specifies how many samples away from the sample containing the specified FLOW_TO_POS that the flow can be and still be considered to go to that point. The default is FLOW_TO_ POS_THRESH=**1**.

- **FLOW_TO_POS_PROJ** - special Projection Specification parameter providing the projection of the FLOW_TO_POS.

- **FILLED_DEM_FILENAME** - specifies the full path and name of a GMG (Global Mapper Grid) file to save the depression filled DEM to after finding it. The default is to not save the filled DEM to a GMG file.

- **GEN_FLOW_DIR_POINTS** - specifies whether or not a separate layer containing a point feature with the flow direction and accumulation at each sample location. Add GEN_FLOW_DIR_POINTS=YES to create the layer. The symbol can be set with the FLOW_DIR_SYMBOL parameter.

- **FLOW_DIR_SYMBOL** - specifies the name of the point symbol to use for the flow direction points. The specified symbol will be rotated to show the flow direction. The default is the medium black arrow.

- **SHOW_FLOW_ACCUM** - specifies whether or not the flow accumulation grid should be saved as a new layer. Use SHOW_FLOW_ACCUM=**YES** to enable saving the flow accumulation to a new grid layer.

- **SPATIAL_RES** - specifies spacing of grid points used to calculate the watershed. A smaller grid spacing results in higher fidelity, but the calculation process will take longer. Typically you'll want to use the default value which is the minimum spatial resolution of all loaded data. Should be formatted as *x_res-olution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to use a grid with a 30 meter spacing, the parameter/value pair would look like `SPATIAL_RES=30.0,30.0`.

- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will auto-matically be converted to the current view/export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.

- **SAMPLING_METHOD** - specifies the sampling method to use when sampling the terrain layers involved in the watershed calculation. The following values are supported:
    - **DEFAULT** - Use either automatic resampling based on export or layer resampling, depending on set-ting of global flag about whether to resample on export
    - **AUTO** - Automatically select a resampling method based on how the export resolution and bounds compare to the original layout for a layer. For example if you export to a lower resolution a box aver-ager of appropriate size may be used automatically
    - **LAYER** - Use the sampling method set for each layer
    - The list of SAMPLING_METHOD values for the IMPORT command can also be specified to use a par-ticular sampling method for all layers being exported/
    - **Shared IMPORT SAMPLING_METHOD values**

      See "SAMPLING_METHOD (elevation and raster only) - specifies the sampling method to use when resampling this layer. " on page 39

- **FILL_GAPS** - specifies that small gaps in between and within the data sets being used to generate the watershed will be filled in by interpolating the surrounding data to come up with an elevation for the point in question. This option is on by default, specify FILL_GAPS=**NO** to turn off.

- **LAYER_DESC** - specifies the name to assign to this layer. If no layer description is provided, the default name of "**GENERATED WATERSHED**" will be used.

- **MIN_STREAM_LEN** - specifies the minimum length in meters of a stream segment at the start of a stream network. Any streams shorter than this that have nothing flowing into them will be discarded. The default is to keep all streams regardless of length.

- **KEEP_ZERO_AT_ZERO** - specifies whether or not to modify 0 (typically ocean) values to allow flow to con-tinue across expanses of 0 elevation. The default is to enable this so 0 values don't change. If you want to model flow across 0 surfaces, add KEEP_ZERO_AT_ZERO=**NO**.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

GENERATE_WATER_RISE

Calculates the water rise from a given area or from a specific elevation.

- **FILENAME**- filename of the terrain layer(s) to process. If an empty value is passed in, all loaded terrain data will be used. This parameter can be listed more than once to specify multiple input files, like `FILENAME-E="FILENAME_1" FILENAME="FILENAME_2"`.

- **ELEVATION_TO_RISE_FROM** -Specify an elevation that the water rise will calculated from.

- **WATER_RISE_POLY_FILE** - Specify the file name for area feature(s) that water rise should be calculated from. Typically this would be a lake or pond.

- **MAX_DEPTH**- specify the height of the water rise and the depression fill depth. This can be height above a specific elevation using ELEVATION_TO_RISE_FROM or height above an area.

- **ELEV_UNITS**- specify the units for elevation values in the command

Shared Parameters

- **SPATIAL_RES** - specifies spacing of grid points used to calculate the watershed. A smaller grid spacing results in higher fidelity, but the calculation process will take longer. Typically you'll want to use the default value which is the minimum spatial resolution of all loaded data. Should be formatted as *x_res-olution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to use a grid with a 30 meter spacing, the parameter/value pair would look like `SPATIAL RES=30.0,30.0`.

- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will auto-matically be converted to the current view/export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.

- **SAMPLING_METHOD** - specifies the sampling method to use when sampling the terrain layers involved in the watershed calculation. The following values are supported:
  - **DEFAULT** - Use either automatic resampling based on export or layer resampling, depending on set-ting of global flag about whether to resample on export
  - **AUTO** - Automatically select a resampling method based on how the export resolution and bounds compare to the original layout for a layer. For example if you export to a lower resolution a box aver-ager of appropriate size may be used automatically
  - **LAYER** - Use the sampling method set for each layer
  - The list of SAMPLING_METHOD values for the IMPORT command can also be specified to use a par-ticular sampling method for all layers being exported/
  - **Shared IMPORT SAMPLING_METHOD values**

    See "SAMPLING_METHOD (elevation and raster only) - specifies the sampling method to use when resampling this layer. " on page 39

- **FILL_GAPS** - specifies that small gaps in between and within the data sets being used to generate the watershed will be filled in by interpolating the surrounding data to come up with an elevation for the point in question. This option is on by default, specify FILL_GAPS=**NO** to turn off.

- **LAYER_DESC** - specifies the name to assign to this layer. If no layer description is provided, the default name of "**Watershed**" will be used.

- **KEEP_ZERO_AT_ZERO** - specifies whether or not to modify 0 (typically ocean) values to allow flow to con-tinue across expanses of 0 elevation. The default is to enable this so 0 values don't change. If you want to model flow across 0 surfaces, add KEEP_ZERO_AT_ZERO=**NO**.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

SAMPLES

Calculate the water rise from the area defined by "riverarea.shp" if the water was to increase 20 ft.

```
GLOBAL_MAPPER_SCRIPT VERSION="1.00" ENABLE_PROGRESS=NO
IMPORT FILENAME="CROPPED_DEM.dem" LAYER_DESC="SOURCE_LAYERS"
IMPORT FILENAME="riverarea.shp" LAYER_DESC="POLYFILE"
GENERATE_WATER_RISE MAX_DEPTH=20 WATER_RISE_POLY_FILE="POLYFILE" ELEV_UNITS=FEET
EXPORT_VECTOR GEN_PRJ_FILE=YES FILENAME="outwaterrise_elev860.shp" \
        TYPE=SHAPEFILE SHAPE_TYPE=AREAS
UNLOAD_ALL
```

Calculate a 2 ft water rise from the given elevation of 860 ft.

```
GLOBAL_MAPPER_SCRIPT VERSION="1.00" ENABLE_PROGRESS=NO
IMPORT FILENAME="CROPPED_DEM.dem" LAYER_DESC="SOURCE_LAYERS"
GENERATE_WATER_RISE MAX_DEPTH=2 ELEVATION_TO_RISE_FROM=860 ELEV_UNITS=FEET
EXPORT_VECTOR GEN_PRJ_FILE=YES FILENAME="outwaterrise_elev860.shp" \
        TYPE=SHAPEFILE SHAPE_TYPE=AREAS
UNLOAD_ALL
```

GENERATE_VIEWSHED

The GENERATE_VIEWSHED command allows you to perform a view shed analysis using loaded elevation grid data with a user-specified transmitter location, height, and radius. All areas within the selected radius that have a clear line of sight to the transmitter are colored with a user-specified color.

The parameters are:

- **LAYER_DESC** - specifies the name to assign to the view shed layer. If no layer description is provided, the default name of "**GENERATE_VIEWSHED Output**" will be used.

- **XMIT_POS** - indicates the transmitter location in the current projection system. This is a required parameter. This should be formatted as XMIT_POS=*"x_easting_longitude,y_northing_latitude"*

- **XMIT_HEIGHT** - specifies the height above ground or sea level for the transmitter that the view shed analysis will be simulating. The value must contain the height and a units abbreviation, e.g., XMIT_HEIGHT="5 ft". **This is a required parameter.**

- **XMIT_HEIGHT_ABOVE_SEA_LEVEL** - indicates whether the XMIT_HEIGHT value is above sea level or above ground. Use XMIT_HEIGHT_ABOVE_SEA_LEVEL=**YES** to specify that the transmitter height is relative to sea level. The default is "**NO**", i.e., the transmitter height is above ground.

- **RECEIVER_ELEV_TYPE** - specifies the method used to define the receiver height. It can be one of the following values:
    - "**EXPLICIT_HEIGHT**" - If RECEIVER_ELEV_TYPE is not specified this is the default.
    - "**XMIT_ANGLE**"
    - "**XMIT_ANGLE_RANGE**"

- **RECEIVER_HEIGHT** - required parameter that specifies the receiver height when the type is "EXPLICIT_ HEIGHT". This is the minimum height above the ground or sea level from which the transmitter must be visible for the point to be considered visible. The value must contain the height and a units abbreviation, e.g., RECEIVER_HEIGHT="8 m".

- **RECEIVER_ANGLE** - Optionally, you can also specify that the receiver elevation should be calculated based on an elevation angle relative to the horizon from the transmitter. This is useful if you have something like a radar dish that points up at some angle and you want to see where the signal can be seen. Specify the value in degrees, from *0 to 360*. RECEIVER_ANGLE is a required parameter when the type is "XMIT_ANGLE" or "XMIT_ANGLE_RANGE".

- **RECEIVER_ANGLE_END** - specifies the other end of a transmission angle range for a beam transmitted from the transmitter. Then the view shed will depict where that beam would hit the terrain surface (or some user-specified distance above the surface). RECEIVER_ANGLE_END is a required parameter when the type is "XMIT_ANGLE_RANGE". Specify the value in degrees, from *0 to 360*. It must be greater than the value used for RECEIVER_ANGLE.

- **RECEIVER_HEIGHT_ANGLE** - indicates the receiver height (in meters) to use when checking with restricted transmission angle. This parameter is optional, and is only used when RECEIVER_ELEV_TYPE is XMIT_HEIGHT_RANGE.

- **RECEIVER_HEIGHT_ABOVE_SEA_LEVEL** - indicates whether the RECEIVER_HEIGHT or RECEIVER_ HEIGHT_ANGLE value is above sea level or above ground. Use RECEIVER_HEIGHT_ABOVE_SEA_LEVEL=YES to specify that the receiver height is relative to sea level. The default is "NO", i.e., the receiver height is relative to the ground.

- **RADIUS** - specifies how far in each direction from the transmitter to check for visibility. Typically you'd want to set this to the effective range of your transmitter. The value must contain the distance and a units abbreviation, e.g., RADIUS="25 km". This is a required parameter.

- **RADIUS_MIN** - If you want to ignore areas close to the transmitter, use RADIUS_MIN to specify a minimum view radius value. The value must contain the distance and a units abbreviation, e.g., RADIUS_MIN="1 km". This is an optional parameter. The default is **0**, which includes everything from the transmitter out to the selected view radius.

- **START_ANGLE** - The GENERATE_VIEWSHED command allows the user to limit the view shed to a particular subsection of the complete radial area. The START_ANGLE specifies the cartographic angle, in degrees, at which the radial subregion begins. This angle is a cartographic angle, meaning that 0 degrees is north and angle increases clockwise. For example, to define an arc that starts due south, use START_ANGLE="180". This parameter is optional. The default is to perform the analysis on the full radial area.

- **SWEPT_ANGLE** - specifies the number of degrees clockwise to include in the view shed. For example, if the transmitter being analyzed sweeps an arc from due south to due west, use `START_ANGLE="180" SWEPT_ ANGLE="90"`. SWEPT_ANGLE is an optional parameter. The default is **360** degrees.

- **USE_EARTH_CURVATURE** - indicates whether or not to take the curvature of the earth into account while performing the view shed analysis. Use USE_EARTH_CURVATURE="**YES**" to specify that you want to use earth curvature. This parameter is optional; the default is "**NO**".

- **ATMOSPHERIC_CORRECTION** - In addition, when earth curvature is being used, use ATMOSTPHERIC_ CORRECTION specify an atmospheric correction value to be used. The atmospheric correction value is useful when determining the view shed for transmitting waves whose path is affected by the atmosphere. For example, when modeling microwave transmissions, `ATMOSPHERIC_CORRECTION="1.333"` is typically used to emulate how microwaves are refracted by the atmosphere. This is an optional parameter, with a default of **1.333.**

- **CREATE_COVERAGE_AREAS** - specifies whether or not view shed coverage area (polygon) features should be generated for those areas that are visible. These generated area features then behave just like any other vector feature and can be exported to vector formats, like Shapefiles, for use in other software. Use CREATE_COVERAGE_AREAS="**NO**" to disable this; the default is "**YES**".

- **SHOW_HIDDEN_AREAS** - indicates whether or not the generated view shed will cover those areas that would NOT be visible, rather than those that would be visible from the transmitter location. Use SHOW_ HIDDEN_AREAS="**YES**" to enable this functionality. The default is to show visible areas.

- **TREAT_INVALID_AS_ZERO** - indicates that invalid or missing values should be treated as zero. The default is "**YES**". Use TREAT_INVALID_AS_ZERO="**NO**" to turn this off.

- **COLOR** - specifies the color to use to display view shed areas. Use an *RGB(red,green,blue)* specification, for example, `COLOR="RGB(0,255,0)"` will create green areas.

- **FRESNEL_FREQ** - allows you to have the view shed analysis also check that a certain portion (FRESNEL_ PCT_CLEAR) of the first Fresnel zone for a transmission of a particular frequency is clear. Use FRESNEL_ FREQ to specify the frequency, in GHz. For example, `FRESNEL_FREQ="2.4"` specifies a frequency of 2.4 GHz.

- **FRESNEL_PCT_CLEAR** - specifies the percent of the first Fresnel zone that must be clear. The typical standard is that good visibility requires that at least 60% (the default) of the first Fresnel zone for the specified frequency be clear of obstructions. Specify the value as a percent, e.g., `FRESNEL_PCT_CLEAR="70"` will use 70%. This is an optional parameter, which will only be used if FRESNEL_FREQ is also specified.

- **FRESNEL_PCT_CLEAR_MAX** - the maximum percent of 1st Fresnel zone that must be clear (default is **100.0**). If you specify a maximum Fresnel zone percentage clear other than 100%, only those locations where the minimum percentage of the 1st Fresnel zone that is clear is between your specified percentages will be marked as visible. This is an optional parameter, which will only be used if FRESNEL_FREQ is also specified.

- **PATH_LOSS_FREQ** - specifies the signal frequency in GHz for the free space path loss calculation. This allows you to display the power at any given location taking free space path loss into account.

- **PATH_LOSS_TOTAL_POWER** - specifies the total power in dB from the rest of the link budget (i.e. transmission power plus antenna gain minus any other power losses excluding free space path loss).

- **GEN_POWER_GRID** - indicates whether or not to create a grid of the remaining power at each location. Then as you move the cursor over the view shed you can see the remaining power. In addition the view shed will get more transparent as the signal power gets lower.

- **USE_VECTOR_HEIGHTS** - indicates whether or not loaded vector data with elevation values should be considered when performing the view shed analysis. This allows you to use things like buildings, fence lines, towers, etc. to block portions of the view, creating a more realistic view shed. The default is to not use vector data. Use USE_VECTOR_HEIGHTS="**YES**" to specify that you want to use heights from vector data.

- **VECTOR_AREAS_HIDDEN** - specifies that any locations within an obstruction area will be marked as hidden, rather than only those that actually would be hidden. The default is "**NO**".

- **VECTOR_HEIGHTS_ABOVE_SEA_LEVEL** - specifies whether the elevation values stored with vector features are relative to the ground or relative to mean sea level. Typically heights for vector features are specified relative to the ground. If any area features are included and their heights are relative to the ground, the obstruction heights within those areas will be increased by the specified amount, but any receiver heights will still be based on the terrain. This makes things like wooded areas very easy to model. The default is "NO" (heights are relative to the ground). To specify that vector heights should be relative to sea level, use VECTOR_HEIGHTS_ABOVE_SEA_LEVEL="**YES**"

- **FIX_INVALID** - indicates whether or not to automatically detect and fix invalid coverage polygons. Use FIX_INVALID="**YES**" to detect and fix invalid polygons. The default is "**NO**".

GENERATE_PATH_PROFILE

The GENERATE_PATH_PROFILE command allows for the saving of a 3D path profile to an ASCII XYZ file. This command uses loaded elevation data to generate a list of the 3D coordinates between two given points in the given elevation units. The following parameters are supported by the command.

- **FILENAME** - full path to XYZ ASCII file to save the data to
- **ELEV_UNITS** - specify elevation units to use in export
    - **FEET** - export in US feet
    - **METERS** - export in meters
- **POINT_COUNT** - specifies the number of points to generate in the path. *This must be at least 2*. For example, to create 1000 points, use POINT_COUNT=1000. You can use the POINT_SPACING parameter rather than this to specify how far apart sample points should be.
- **POINT_SPACING** - specifies the point spacing in meters to use between sample points along the path. For example, to create points spaced 10 meters apart, use POINT_SPACING=10.0.
- **START_POS** - specifies the start position for the path profile. The coordinates must be given in the current global coordinate system. For example, if UTM if the current projection, you might specify and easting/northing as follows: START_POS=480000,4310000.
- **END_POS** - specifies the end position for the path profile. The coordinates must be given in the current global coordinate system. For example, if UTM if the current projection, you might specify and easting/northing as follows: START_POS=480000,4310000.
- **ADD_LAND_USE_CODES** - specifies whether to query loaded LULC data sets for the land use code at each point and to include that land use code after the elevation. Use ADD_LAND_USE_CODES=**YES** to turn on adding land use codes for each point.
- **APPEND_TO_FILE** - specifies that the elevations between the start and end locations should be appended to the file specified if it already exists rather than a new file being created. Use APPEND_TO_FILE=**YES** to enable.
- **ADD_BLANK_LINE** - specifies that a blank line will be added to the file if APPEND_TO_FILE=YES is added to the command and the file was not empty to start with. Use ADD_BLANK_LINE=**YES** to enable adding the blank line.
- **SAVE_DIST_Z_FILE** - specifies that the output file should contain distance and elevation values rather than XYZ coordinate values. Use SAVE_DIST_Z_FILE=**YES** to enable this option.

GENERATE_ELEV_GRID

The GENERATE_ELEV_GRID command allows for the generation of a gridded elevation layer using loaded 3D vector data. The following parameters are supported by the command as well as the display option parameters supported by the IMPORT command.

- **FILENAME** - filename of the layer(s) to grid. If an empty value is passed in, all loaded vector layers with 3D data will be gridded. This parameter can be listed more than once to specify multiple input files, like FILENAME="FILENAME_1" FILENAME="FILENAME_2". When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center gridded.
- **ELEV_UNITS** - specify elevation units to use for new grid
    - **FEET** - US feet
    - **DECIFEET** - 10ths of US feet
    - **METERS** - meters
    - **DECIMETERS** - 10ths of meters
    - **CENTIMETERS** - centimeters

- **GRID_ALG** - specify gridding algorithm to use
  - **TIN** - triangulate 3D data and grid it. This is the default and allows use of lines and areas as constraints.
  - **BIN_MIN** - uses the minimum value within a bin of size GRID_BIN_SIZE. Requires a Lidar module license . Lidar data will be binned, and 3D line an area features will be used as grid constraints/ break-lines.
  - **BIN_AVG** - uses the average value within a bin of size GRID_BIN_SIZE. Requires a Lidar module license . Lidar data will be binned, and 3D line an area features will be used as grid constraints/ break-lines.
  - **BIN_MAX** - uses the maximum value within a bin of size GRID_BIN_SIZE. Requires a Lidar module license. Lidar data will be binned, and 3D line an area features will be used as grid constraints/ break-lines.
- **SPATIAL_RES** - specifies spacing of grid points to use in generated grid. A smaller grid spacing results in higher fidelity, but larger, elevation grids. Should be formatted as *x_resolution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to use a grid with a 30 meter spacing, the parameter/ value pair would look like `SPATIAL_RES-S=30.0,30.0`. If you do not provide a SPATIAL_RES value, a good default for the input data will be chosen, so in most cases it is best just to leave this off.
- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will automatically be converted to the current view/ export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.
- **GRID_BIN_SIZE** - if using a bin-based GRID_ALG for Lidar data this specifies how many native spacings in size to make each bin. For example a value of `GRID_BIN_SIZE="3.0"` would make each square bin 3 times the calculated native spacing of the point data. You can specify a bin size in meters by using the SPATIAL_ RES_METERS parameter or a *negative* GRID_BIN_SIZE (like `GRID_BIN_SIZE="-0.5"` for 0.5 meter spacing.
- **LAYER_DESC** - specifies the name to assign to this layer. If no layer description is provided, a default name will be assigned.
- **NO_DATA_DIST_MULT** - specifies how far from an actual data point a grid cell has to be before it is treated as a no data value. This number is given as a multiple of the diagonal size of a single grid cell as nominally determined by the gridding algorithm or specified with the SPATIAL_RES parameter. A value of **0** (the default) means that all points should be considered as valid.
- **GRID_FILL_TO_BOUNDS** - specifies that the grid values should be filled out to the entire bounds of the gridded data rather than just to the convex hull of the data being gridded. Use GRID_FILL_TO_ BOUNDS=**YES** to enable this.
- **GRID_FLATTEN_AREAS** - specifies that area features with elevation values should be flattened to the elevation of the area features. Use GRID_FLATTEN_AREAS=**NO** to disable this.
- **GRID_HEIGHTS_RELATIVE** - specifies that the elevation values for the input vector features should be treated as relative to any loaded terrain data rather than as absolute elevation values. This useful for things like trees or buildings where you have a height above the ground rather than an absolute height. Use GRID_HEIGHTS_RELATIVE=**YES** to enable this.
- **GRID_SAVE_TIN** - specifies that the triangulated irregular network (TIN) for the grid operation should be saved as a new separate vector layer consisting of triangular 3D area features. Use GRID_SAVE_ TIN=**YES** to enable this.
- **GRID_USE_CONSTRAINTS** - specifies that 3D line and area features should be treated as constraints (breaklines) during the gridding process. Use GRID_USE_CONSTRAINTS=**YES** to enable this. GRID_USE_ CONSTRAINTS=**NO** will disable using 3D line and area features in the grid generation. The method for applying the constraints or breaklines depends on the GRID_ALG. The binning methods use a soft edge value of 2 samples.
- **GRID_IGNORE_ZERO** - specifies that features with an elevation of 0.0 will not be used during the gridding process.
- **GRID_TYPE** - specifies what values should be gridded for a bin-based grid. The following values are supported:
  - **ELEVATION** - **default**, grids the elevation values for each point
  - **INTENSITY** - grids the intensity value for each point
  - **HEIGHT_ABOVE_GROUND** - grids the height above ground for each point
  - **NDVI** - grids the calculate NDVI (normalized difference vegetation index) for 4-band (RGB+NIR) Lidar points
  - **NDWI** - grids the calculate NDWI (normalized difference water index) for 4-band (RGB+NIR) Lidar points

- **Lidar Point Filter Parameters**

  See also "Lidar Advanced Filter Options" on page 79

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

GENERATE_POINTS_FROM_ELEV_GRID

Creates a point feature at the center of each cell in the specified elevation grid layer(s).

The available parameters for the command include:

- **FILENAME** - filename or layer description of the layer(s) to process. If an empty value, or "*", is passed in, all loaded elevation grid layers will be included, and a separate output layer will be produced for each input layer. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`.

- **LAYER_DESC** - specifies the name to assign to the output layer. If no layer description is provided, a default name will be assigned. If more than one layer is being processed, this will be used as a suffix on the automatically generated layer name.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

Example:

```
GENERATE_POINTS_FROM_ELEV_GRID FILENAME="elev_grid.gmg" \
LAYER_DESC="grid_points" \
POLYGON_CROP_FILE="CropAreas.shp" POLYGON_CROP_USE_ALL=YES
```

Lidar Analysis

LIDAR_CLASSIFY

The LIDAR_CLASSIFY command allows for automatically identifying and classifying ground-shot or building/high vegetation points from Lidar point clouds. The following parameters are supported by the command:

- **FILENAME** - filename or description of loaded layer(s) to classify Lidar points in. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`.

- **TYPE** - specifies what type(s) of points to classify. If you don't provide a TYPE parameter only ground points will be classified. To classify multiple types, provide a comma-separated list, like `TYP-E="GROUND,NONGROUND,POWERLINE"`. The following values are valid:
  - **GROUND** - classify ground shot points.
  - **NONGROUND** - classify building/high vegetation points. Note you must already have classified ground points for the building/tree classification to work properly.
  - **POWERLINE** - classify powerline vegetation points. Note you must already have classified ground points for the powerline classification to work properly.
  - **NOISE** - classify high and low noise points.
  - **HIGH_NOISE** - classify high noise points.
  - **LOW_NOISE** - classify low noise points.

Ground Point Classification Options

Use for TYPE=GROUND

- **GRID_BIN_SIZE** - specifies how many native spacings in size to make each bin initially for the algorithm. For example a value of `GRID_BIN_SIZE="3.0"` would make each square bin 3 times the calculated native spacing of the point data. The **default** is GRID_BIN_SIZE="**3.0**" for classifying ground points and GRID_BIN_SIZE="**1.0**" for non-ground (building/tree) points. If you want to specify a spacing in *meters* rather than as a multiple of the native spacing for the point cloud, use a negative value (**-**). For example, to get a spacing of 0.6 meters, use `GRID_BIN_SIZE="-0.6"`.

- **LIDAR_RESET_GROUND** - specifies that any points that are already marked as ground should be reset to unclassified to start the process. Add LIDAR_RESET_GROUND=**YES** to reset the points.

- **LIDAR_CURVATURE** - specifies the minimum height differential (curvature) in meters for the first pass of the MCC algorithm. The **default** is LIDAR_CURVATURE="**0.3**".

- **LIDAR_MAX_HEIGHT_DELTA** - specifies the maximum height difference to consider as still possibly ground when removing points that are likely non-ground (i.e. buildings or vegetation) using a morphological filter before running the MCC algorithm to find ground points. You can provide a value of **0** to skip the morphological filter altogether and just run the MCC algorithm.

- **LIDAR_SLOPE** - specifies the slope in degrees that is close to the expected steepest slope in the region being classified. This is used by the morphological filter which removes likely non-ground points before running the MCC algorithm to find ground points.

Non-Ground (Building/Tree) Point Classification Options

use for TYPE=NONGROUND.

- **GRID_BIN_SIZE** - specifies how many native spacings in size to make each bin initially for the algorithm. For example a value of `GRID_BIN_SIZE="3.0"` would make each square bin 3 times the calculated native spacing of the point data. The **default** is GRID_BIN_SIZE="**3.0**" for classifying ground points and GRID_BIN_

SIZE="**1.0**" for non-ground (building/tree) points. If you want to specify a spacing in *meters* rather than as a multiple of the native spacing for the point cloud, use a negative value (**-**). For example, to get a spacing of 0.6 meters, use GRID_BIN_SIZE="-0.6".

- **LIDAR_RESET_NON_GROUND** - specifies that any points that are already marked as one of the types being classified should be reset to unclassified at the start of the operation. Add LIDAR_RESET_ NONGROUND=**YES** to reset the points.

- **LIDAR_MIN_HEIGHT** - specifies the minimum height above ground that a point has to be in order to consider it as a possible building or high vegetation point.

- **LIDAR_PLANE_MAX_OFFSET** - specifies the maximum RMSE (root mean square error) in meters from a best-fit local plane that the points in a small region all have to be within in order to consider the region a potential planar (building) region. The **default** is LIDAR_PLANE_MAX_OFFSET=**0.08** (8 cm). If you have lower resolution data you might need to bump this up a bit. A good value should be at least a couple of times the absolute error in elevation for the data set.

- **LIDAR_PLANE_MAX_ANGLE** - specifies the maximum angle (in degrees) between adjacent best-fit planes such that they can still be considered part of the same plane when identifying flat building surfaces.

- **LIDAR_TREE_MAX_OFFSET** - specifies the minimum RMSE (root mean square error) in meters from a best-fit local plane that the points in a small region all have to be within in order to consider the region a potential vegetative region. The **default** is LIDAR_TREE_MAX_OFFSET=**0.15** (15 cm). If you have lower resolution data you might need to bump this up a bit. This value *must be larger than the* LIDAR_PLANE_ MAX_OFFSET *value*.

Powerline Point Classification Options

use for TYPE=POWERLINE.

- **POWERLINE_BIN_SIZE** - specifies the size of each bin (meters per edge) when evaluating points to see if they are clustered as needed for the powerline classification algorithm. Typically you will leave this at the **default** setting of 1 meter bins (i.e. POWERLINE_BIN_SIZE="**1.0**"). You might use slightly larger bins for data that is lower density (around 20 points / sq m), up to 2 meter bins.

- **LIDAR_RESET_NONGROUND** - specifies that any points that are classified as a powerline-related type will be reset to unclassified at the start of the operation and that other already classified non-ground points will be considered as possible powerlines. Add LIDAR_RESET_NONGROUND=**YES** to reset the points.

- **LIDAR_MIN_POWERLINE_HEIGHT** - specifies the minimum height above ground (in meters) that a point has to be in order to consider it as a possible powerline point.

- **POWERLINE_MAX_DIST_FROM_LINE** - specifies the maximum distance (in meters) from the best-fit 3D line of points with similar elevations in a bin that any points can be and still be considered powerlines.

- **POWERLINE_MAX_VERT_DIFF_PER_M** - specifies the maximum difference in elevation allowed per meter to consider points as possibly part of the same powerline segment. The **default** value is **0.5**m, which allows for a change in elevation of 0.5m over a 1m distance between points. You might specify a slightly larger value if your data is noisy.

Noise Point Classification Options

use for TYPE=NOISE.

- **NOISE_BIN_SIZE** - specifies how many native spacings in size to make each bin initially for the noise classification algorithm. The noise classification algorithm identifies points that are far outside the normal range in a local area of the point cloud. This value will typically be large, like NOISE_BIN_SIZE=128, to consider local areas that are 128 times the nominal point spacing in each direction.

- **LIDAR_RESET_NOISE** - specifies that any points that are already marked as noise should be reset to unclassified to start the process. Add LIDAR_RESET_NOISE=**YES** to reset the points.

- **LIDAR_RESET_CLASSIFIED** - controls whether or not points that are already classified can be identified and marked as noise points. Add LIDAR_RESET_UNCLASSIFIED=**NO** to only check unclassified points for noise.

- **NOISE_STD_DEV** - specifies how many standard deviations above or below the mean for an area that a point elevation needs to be in order to be considered noise. **Default** is NOISE_STD_DEV=**3.0**.

- **LIDAR_ELEV_RANGE** - specifies that any points with elevations outside of a specified range should be marked as high or low noise. To specify a range of acceptable values in meters, specify the *minimum*

allowed elevation followed by a comma and the maximum allowed elevation. For example, if you want to mark everything outside the range 50m - 150m as noise, you could add `LIDAR_ELEV_RANGE="50,150"`.

- **LIDAR_HEIGHT_RANGE** - specifies that any points with height above ground outside of a specified range should be marked as high or low noise. To specify a range of acceptable values in meters, *specify the minimum allowed height followed by a comma and the maximum allowed height*. For example, if you want to mark everything more than 2 meters below the ground surface as low noise and everything more than 500m above the ground surface as high noise, use `LIDAR_HEIGHT_RANGE="-2,500"`. The **default** value is LIDAR_HEIGHT_RANGE="**-2,200**".

- **CHANGE_CLASS** - specifies that any points that are identified as noise should have their classification changed. This defaults to on, so add CHANGE_CLASS=**NO** to disable.

- **MARK_WITHHELD** - specifies that any points that are identified as noise should be marked as withheld. Use MARK_WITHHELD=**YES** to enable this behavior. By default this will be the opposite of the CHANGE_CLASS value.

- **DELETE_FEATURES** - specifies that any points that are identified as noise should be marked as deleted. This defaults to the same as the MARK_WITHHELD value. Explicitly add DELETE_FEATURES=**YES** to always mark noise as deleted or DELETE_FEATURES=**NO** to never mark as deleted.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Lidar Advanced Filter Parameters**

  See also "Lidar Advanced Filter Options" on page 79

LIDAR_COMPARE

The LIDAR_COMPARE command allows for comparing the elevations from loaded Lidar point clouds to loaded 3D control points. You can also adjust the Lidar points to match the control points. The following parameters are supported by the command:

- **FILENAME** - filename or description of loaded layer(s) of Lidar point clouds to compare to control point(s). This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`. If not specified all loaded Lidar clouds will be used.

- **POINT_FILENAME** - filename or description of loaded layer(s) with the 3D control point features. If not specified all loaded 3D point features will be used.

- **REPORT_FILENAME** - if specified, the name of the text file (CSV format) to save a report to with each control point listed along with information about the elevation difference at the point as well as the overall RMSE in meters

- **FIT_POINTS** - if FIT_POINTS=**YES** is provided, the elevation of the Lidar points will be adjusted to match the control point elevations.

SAMPLE

```
LIDAR_COMPARE REPORT_FILENAME="%SCRIPT_FOLDER%report.csv" FIT_POINTS=YES
```

LIDAR_EXTRACT

The LIDAR_EXTRACT command allows for automatically extracting building outlines, tree points/outlines, and linear powerlines from classified Lidar point clouds. Building extraction requires classified building points, tree extraction requires classified high vegetation points, and powerline extraction requires classified powerline points. The following parameters are supported by the command:

- **FILENAME** - filename or description of loaded Lidar layer(s) to extract from. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`.

- **GRID_BIN_SIZE** - specifies how many native spacings in size to make each bin initially for the algorithm. For example a value of `GRID_BIN_SIZE="3.0"` would make each square bin 3 times the calculated native spacing of the point data. The **default** is GRID_BIN_SIZE="**0.5**" which queries the data at half of the native spacing. If you want to specify a spacing in *meters* rather than as a multiple of the native spacing for the point cloud, use a negative value. For example, to get a spacing of 0.6 meters, use `GRID_BIN_SIZE="-0.6"`.
- **TYPE** - specifies what type of features to extract. If you don't provide a TYPE parameter all types will be extracted. The following values are valid:
  - **ALL** - extract all types
  - **BUILDING** - extract building/roof outlines. Note you must already have classified building points for the building extraction to work.
  - **TREE** - extract tree points/outlines. Note you must already have classified high vegetation points for the tree extraction to work.
  - **POWERLINE** - extract powerline features. Note you must already have classified powerline points for the line extraction to work.

Building Extraction Options

Use for TYPE=BUILDING.

- **CREATE_SIDE_WALLS** - specifies whether or not side wall areas will be created extended from the roof polygon edges to the ground height. Use CREATE_SIDE_WALLS=**YES** to enable. If not enabled, the roof polygons will be marked to extrude to the ground when displaying in the 3D viewer.
- **CREATE_SEP_ROOF_AREAS** - specifies that separate roof areas will be created where the direction and/or Z value of the roof plane significantly changes between adjacent roof cells. The LIDAR_PLANE_MAX_OFFSET and LIDAR_PLANE_MAX_ANGLE parameters control how large of an offset is allowed before breaking the roof. Use CREATE_SEP_ROOF_AREAS=**YES** to enable.
- L**IDAR_PLANE_MAX_OFFSET** - specifies the maximum RMSE (root mean square error) in meters from a best-fit local plane that the points in a small region all have to be within in order to consider the region a potential planar (building) region. The **default** is LIDAR_PLANE_MAX_OFFSET=**0.08** (8 cm). If you have lower resolution data you might need to bump this up a bit. A good value should be at least a couple of times the absolute error in elevation for the data set.
- **LIDAR_PLANE_MAX_ANGLE** - specifies the maximum angle (in degrees) between adjacent best-fit planes such that they can still be considered part of the same plane when identifying flat building surfaces.
- **SIMPLIFICATION** - specifies the simplification multiplier to use when simplifying the building polygons.

Tree Extraction Options

Use for TYPE=TREE.

- **CREATE_TREE_POLYS** - specified that approximate tree outline polygons should be created. Add CREATE_TREE_POLYS=**YES** to enable.
- **LIDAR_MIN_HEIGHT** - specifies the minimum height above ground (in meters) for a tree crown point to create a point at the tree crown location.
- **LIDAR_TREE_MIN_SPREAD** - specifies the minimum spread/width (in meters) which is the minimum width of a tree
- **LIDAR_TREE_MAX_SPREAD** - specifies the maximum spread/width (in meters) which is the maximum width of a tree

Powerline Extraction Options

Use for TYPE=POWERLINE.

- **POWERLINE_MAX_DIST_FROM_LINE** - specifies the maximum distance (in meters) from the best-fit 3D line of points for a connected powerline segment that the points in a new candidate segment can be and still be connected.
- **POWERLINE_MAX_ANGLE_DELTA** - specifies the maximum difference in angle (in degrees) from a straight line that the points in a new candidate segment can be and still be connected.
- **POWERLINE_MIN_LEN** - specifies the minimum total length in meters that a connected powerline must be in order to be kept

- **POWERLINE_MAX_VERT_DIFF_PER_M** - specifies the maximum difference in elevation allowed per meter to allow when connecting short segments to a longer continuous powerline. The **default** value is **0.5**m, which allows for a change in elevation of 0.5m over a 1m distance between points. You might specify a slightly larger value if your data is noisy.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Lidar Advanced Filter Options**

  See aso "Lidar Advanced Filter Options" below

Lidar Advanced Filter Options

Lidar filter parameters can be applied to operations that process lidar data, including GENERATE_ELEV_GRID, EXPORT_VECTOR, LIDAR_CLASSIFY and LIDAR_EXTRACT

The available lidar filter parameters include:

- **LIDAR_ELEV_RANGE** - specifies the range of elevations to include in the grid in meters. By default all elevations are gridded, but if you want to restrict values to say 50m - 150m, you could add `LIDAR_ELEV_ RANGE="50,150"`.

- **LIDAR_SCAN_ANGLE_RANGE** - specifies the range of scan angles to include in the grid in degrees. By default all scan angles are gridded, but if you want to restrict the grid to only those points with scan angles between 0 and 30 degrees, you could add `LIDAR_SCAN_ANGLE_RANGE="0,30"`.

- **LIDAR_FILTER** - specifies a comma-separated list of Lidar class numbers to export. Provide a minus sign to remove the type from the filter rather than add it. The filter starts off with nothing in it if you provide a LIDAR_FILTER string, but you can add **ALL** to enable everything or **NONE** to clear the filter, then add or remove stuff after that. For example, to specify a class filter with only types 2 and 3 enabled, use `LIDAR_ FILTER="NONE,2,3"`. To get one with everything but classes 2 and 3, use `LIDAR_FILTER="ALL,-2,-3"`. If no LIDAR_FILTER is provided then all types currently enabled in the shared global Lidar filter are used.

- **LIDAR_RETURN_FILTER** - specifies a comma-separated list of Lidar return types to enable or disable. Provide a minus sign to remove the type from the filter rather than add it. The filter starts off with the current filter settings, but you can add ALL to enable everything or NONE to clear the filter, then add or remove stuff after that. For example, to specify a return filter with only unknown and first returns, use `LIDAR_RETURN_FILTER="NONE,0,1"`. To get one with everything but the first return, use `LIDAR_RETURN_ FILTER="ALL,-1"`. The numeric values have the following meanings:
  - **0** - Unknown Returns
  - **1** - First Return
  - **2** - Second Return
  - **3** - Last Return
  - **4** - Single Return
  - **5** - First of Many Returns
  - **6** - Second of Many Returns
  - **7** - Third of Many Returns
  - **8** - Last of Many Returns

- **LIDAR_COLOR_FILTER** - specifies a color to include in the grid. If no value is provided then all colors are gridded. Otherwise, you can provide multiple LIDAR_COLOR_FILTER parameters of the format LIDAR_ COLOR_FILTER=*"RGB(red,green,blue)"* to specify colors to keep. The LIDAR_COLOR_DIST parameter specifies how far from an exact match to a specified color that a point color can be to be kept.

- **LIDAR_DENSITY_RANGE** - specifies the range of point densities in points per square meter to include. Any Lidar points in regions with densities outside the range are ignored. If you use two values then *everything between the values* is used. If only one value is specified then *all points in areas >= to the specified value* are used. For example, `LIDAR_DENSITY_RANGE="1.0"` means that all points in areas with densities of 1.0 points per square meter or higher are used.

- **LIDAR_SOURCE_ID_LIST** - specifies a comma-separated list of point source IDs to keep. If no list is provided all points are kept. For example, to keep just points with a source ID of 5 or 6, use `LIDAR_SOURCE_ ID_LIST="5,6"`.

Edit Vector Data

This set of commands includes many of the tools available in the digitizer, including selection and editing.

EDIT_VECTOR

The EDIT_VECTOR command selects and modifies vector data. You can choose to update area, line, and/ or point features with a single operation. You can also supply multiple COMPARE_STR parameters to apply multiple criteria, all of which must be true, in order to edit a feature.

It allows you to assign feature types (classifications), add/update attributes and display labels, and reshape or delete features based on one or more attribute or label values.

The following parameters are supported by the command:

Specify Data to Edit (By Attribute and/or Bounding Box)

- **FILENAME** - filename of the layer to update. If an *empty value* is passed in, all loaded vector layers will be updated. This parameter can be listed more than once to specify multiple input files, like FILENAM- E="FILENAME_1" FILENAME="FILENAME_2". When running the script in the context of the main map view (including loading a workspace) you can also pass in the value **'USER CREATED FEATURES'** to have the 'User Created Features' layer updated or **'SELECTED LAYERS'** to have any layers selected in the Control Center updated.

- **SHAPE_TYPE** - specifies the vector object type(s) (area, line, and/or point) to edit. If you don't provide a value then all available matching features will be edited. You can specify multiple different object types as a comma-delimited list of the following (like SHAPE_TYPE="AREAS,LINES"):
  - **AREAS** - area features
  - **LINES** - line features
  - **POINTS** - point features

- **COMPARE_STR** - specifies a comparison operation to perform to see if a feature is one that needs to be updated. The format is *<attr_name><operator><attr_value>* where the operator can be "**=**", "**!=**", "**<**", "**<=**", "**>**", or "**>=**". For example, to get all features with CLASS attribute less than 3, use COMPARE_STR- R="CLASS<3".
  - You can also use a wild card symbol ("*") to match any character. For example if you have an attrib- ute named CFCC and you want to match when the value of that attribute starts with an 'A', you can use COMPARE_STR="CFCC=A*" as your parameter. Note that when you use a wild card, only the "=" and "!=" comparison operators will result in a successful match.
  - Instead of testing a specific attribute name, you can test whether any attribute in a feature has a par- ticular value. To do this, use "**<Any Attribute>**" as the attribute name. For example, COMPARE_STR- R="<Any Attribute>=1" will match all features that have an attribute containing the value "1". You can use any operator, and the comparison value can contain wildcards.
  - You can add multiple COMPARE_STR parameters to a single command to combine multiple criteria for your search. See"Attribute Name Values" on page 92 details for other special allowed attribute names. Special attributes can only be used for "**=**" and "**!=**" comparisons.

- **CASE_SENSITIVE** - specifies whether or not text comparisons are case sensitive or not. Use CASE_ SENSITIVE=**YES** to enable, by default comparisons are not case sensitive.

- **ATTR_EXISTS** and **ATTR_MISSING** - specialized instances of COMPARE_STR that allow you to select fea- tures based on whether or not an attribute exists. For example, use ATTR_EXISTS="CFCC" to select features that have an attribute called CFCC, or ATTR_MISSING="CFCC" to select features that do not have an attribute called CFCC. ATTR_EXISTS and ATTR_MISSING parameters can be used more than once in an EDIT_ VECTOR command, and can be combined with COMPARE_STR parameters.

- **COMPARE_OP** - controls how multiple COMPARE_STR, ATTR_EXISTS, and ATTR_MISSING parameters are handled. The default is to require that *all* conditions be met to include a feature (an AND operation), but if you would like to match on a feature if at least one condition is met, use COMPARE_OP=**ANY** to specify that

a logical OR operation should be used rather than an AND operator. You can explicitly add COMPARE_ OP=**ALL** if you want a logical AND operation, but that is the default and it will work that way if you leave it off as well.

- **COMPARE_NUM** - specifies that any comparisons that are done against numeric values should be numeric rather than alphabetic. Use COMPARE_NUM=**YES** to enable.

Specify Layer for Output - Default is Input Layer

- **COPY_TO_NEW_LAYER** - specify COPY_TO_NEW_LAYER=**YES** to have the results (features) of the EDIT_VECTOR command copied to a new layer. If you use this parameter, but do not specify a value, then YES will be assumed.

- **MOVE_TO_NEW_LAYER** - specify MOVE_TO_NEW_LAYER=**YES** to have the results (features) of the EDIT_VECTOR command copied to a new layer, and then deleted from their original layer. If you use this parameter, but do not specify a value, then YES will be assumed.

- **NEW_LAYER_NAME** - Indicates the name of the layer where the features will be created, copied or moved. If a layer with this name already exists, then that layer will be used, otherwise, a new layer will be created.

- **NEW_LAYER_PROJ** - special Projection Specification indicating the projection to be used in the new layer. If this parameter is not specified, then the new layer will use the current global projection.

Attribute and Style Editing

See also "Attribute Management" on page 88 and "DEFINE_LAYER_STYLE" on page 28

- **AREA_TYPE** - specifies the name of the Global Mapper area type to assign to matching area features.

- **LINE_TYPE** - specifies the name of the Global Mapper line type to assign to matching line features.

- **POINT_TYPE** - specifies the name of the Global Mapper point type to assign to matching point features.

- **LIDAR_CLASS** - specifies the name of the Lidar class to apply to all matching Lidar points. This can either be the Lidar class number or the name of the Lidar class displayed in the user interface. For example, to set all points to type 2 (ground), use LIDAR_CLASS=2 or LIDAR_CLASS="Ground". This option requires a Lidar module license.

- **ATTR_VAL** - specifies the attribute value to update and what value to assign to it. The format is *attr_name- e=attr_value*. For example if you want to set the value of an attribute named CFCC to A34, use ATTR_VAL- L="CFCC=A34". You can add multiple ATTR_VAL parameters to a single command to provide multiple attribute/value pairs (or labels) to add/ update. If you would like to update the feature label rather than an attribute, use **<Feature Name>** for your attribute name.

- **ATTR_TO_DELETE** - specifies the name of an attribute to delete from matching features. You can include multiple instances of this attribute to delete multiple values. To remove the feature label rather than an attribute, use **<Feature Name>** for the attribute name.

- **ATTR_TO_KEEP** - specifies the name of an attribute to keep from matching features. You can include multiple instances of this attribute to specify multiple attributes to keep. You can also include wildcard characters, like * and ?, in the attribute name.

- **ATTR_TO_RENAME** - specifies the name of an attribute to rename from matching features and what to rename it to. You can include multiple instances of this attribute to rename multiple attributes. For example, to rename the attribute, CTY to be COUNTY, add the parameter ATTR_TO_RENAME="CTY=COUNTY" to your command.

- **ATTR_TO_COPY** - specifies the name of an attribute to copy the value of an existing attribute or label to. You can include multiple instances of this parameter to copy multiple attributes. For example, to create a new attribute named DISP_LABEL from the feature label, use ATTR_TO_COPY="DISP_LABEL=<Feature Name>" or to create a new attribute named DEPTH from the value of an attribute named Z, use ATTR_TO_COPY- Y="DEPTH=Z".

- **ATTR_REPLACE_STR** - specifies the *name of an attribute to replace text in, the text to be replace, and the new text*. You can include multiple instances of this attribute to replace text in multiple attributes. For example, to replace the text 'Street' with 'St.' in an attribute named 'ROAD_NAME', use ATTR_REPLACE_STR- R="ROAD_NAME=Street=St." in your command. To replace text in the feature label rather than an attribute, use **<Feature Name>** for the attribute name. To replace a new line character, use the escape sequence **\n**.

- **STYLE_ATTR** - provides a style attribute to update. You can include multiple STYLE_ATTR parameters, one for each style attribute pair. See Field Options for a list of the supported style attributes and values. For example to set all matching lines to a solid green pen 3 pixels wide, add the following: `STYLE_ATTR="LINE_STYLE=SOLID" STYLE_ATTR="LINE_WIDTH=3" STYLE_ATTR="LINE_COLOR=RGB(0,255,0)"`

- **ADD_COORD_ATTRS** - specifies that X and Y attributes should be added with the X and Y coordinate values to the attribute list of any matching point features. For matching line features you will get attributes for the start and end points (START_X, START_Y, END_X, and END_Y).

Duplicate Feature Finding

- **DELETE_DUPLICATES** - specifies whether or not to mark matching features that are duplicates (i.e. same coordinates, label, attributes, type, and style) as deleted. Use DELETE_DUPLICATES=**YES** to enable. Note that *this only removes duplicate features that are within the same layer*. Duplicates in different layers will not be deleted.

- **MARK_DUPLICATES** - specifies whether or not to add a DUPLICATE=Y attribute to matching features that are duplicates (i.e. same coordinates, label, attributes, type, and style). Use MARK_DUPLICATES=**YES** to enable. Note that *this only marks duplicate features that are within the same layer*. Duplicates in different layers will not be marked. All features in the duplicate set will be marked with the DUPLICATE=Y attribute except the first one. All of the duplicates will also get a DUP_ID attribute that uniquely identifies the group of duplicates.

- **IGNORE_ATTRS** - specifies when deleting or marking duplicates that only the geometry of the features should be considered and not any attribute values. Add IGNORE_ATTRS=**YES** to make the attributes be ignored. You can require some particular attributes to be equal using the DUPLICATE_ATTR parameter.

- **DUPLICATE_ATTR** - specifies the name of an attribute to that must have the same value in both features being compared when looking for duplicates. You can include multiple instances of this attribute to require multiple attributes to be equal. Special attribute names, like **<Feature Name>**, are recognized for the attribute name. See "Attribute Name Values" on page 92. For example to require the name and an attribute named ROAD_ID to be equal, add `DUPLICATE_ATTR="ROAD_ID"` and `DUPLICATE_ATTR="<Feature Name>"` to your command. This would only be used in conjunction with `IGNORE_ATTRS=YES` otherwise all attributes are required to be equal.

- **SUBSET_IS_DUP** - specifies whether a line that is a subset of another line should be treated as a duplicate of the other line. If you add SUBSET_IS_DUP, any lines that are sub-sets of another line will have a SUBSET=Y attribute added along with the other duplicate information.

Apply Terrain Elevations to Vector Data

- **APPLY_ELEVS** - Enable this option is you want to apply elevation values from loaded terrain layers to vector data. Use APPLY_ELEVS=**YES** to enable

- **CALC_ELEV_SLOPE_STATS** - uses loaded elevation data to calculate elevation and slope statistics for all features that meet the selection criteria. It can be used in combination with the following parameters: IGNORE_LINE_VERTICES, INC_UNIT_SUFFIX, ELEV_ATTR, ELEV_LAYER.

- **ELEV_ATTR** - The name the attribute to save the elevation in when applying to point features. If not provided, the value will be stored in the ELEVATION attribute.

- **ELEV_LAYER** - filename or description of elevation layer(s) to get elevation values from. By **default** all visible terrain layers are used. You can include multiple ELEV_LAYER parameters if you have multiple masks to search. Wildcards (* and ?) are supported. Hidden layers are not considered.

- **IGNORE_LINE_VERTICES** - while computing statistics for a line feature, resample the feature instead of using the features vertices. This parameter works with APPLY_ELEVS or CALC_ELEV_SLOPE_STATS.

- **INC_UNIT_SUFFIX** - specifies whether or not the unit suffix (like 'm' or 'ft') is appended after the raw elevation value for elevations saved to attributes for point features. The default is off, use INC_UNIT_SUFFIX=**YES** to enable

- **REPLACE_EXISTING** - specifies whether we should replace existing elevation values with a new one from terrain layers. This is enabled by default. Use REPLACE_EXISTING=**NO** to disable replacing existing elevations.

- **ADD_EXISTING_ELEV** - if replacing existing elevation values, specifies whether the elevation at each location from terrain layers is added to the existing elevation value rather than directly replacing it. Use ADD_EXISTING_ELEV=**YES** to enable.

Buffer Creation

The following parameters control creation of buffer areas around matching features:

- **BUFFER_DIST** - explicit buffer radius parameter, in meters by default. e.g. `BUFFER_DIST="5.0"`. You can also put units in string (like `BUFFER_DIST="5.0 miles"` or `BUFFER_DIST="500 m"`). Put space between distance and units.
- **BUFFER_DIST_END** - explicit buffer radius parameter for radius at end of *tapered* buffer, in meters by default. e.g. `BUFFER_DIST_END="3.0"`. You can also put units in string (like `BUFFER_DIST="5.0 miles"` or `BUFFER_DIST="500 m"`). Put space between distance and units.
- **BUFFER_ATTR** - name of attribute that determines buffer radius for a feature. One of "BUFFER_ATTR" or "BUFFER_DIST" must be used. e.g. `BUFFER_ATTR="BUFFERSIZE"`
- **BUFFER_ATTR_END** - name of attribute that determines buffer radius at the end of a *tapered buffer* for a feature. One of "BUFFER_ATTR_END" or "BUFFER_DIST_END" must be used to get a tapered buffer
- **BUFFER_ATTR_UNITS** - String representing units to use to interpret value obtained by "BUFFER_ATTR". Accepts "**M**" (meters), "**METERS**", "**FT**", "**FEET**", "**KM**", "**MILES**", "**NM**" (nautical miles). Defaults to meters. e.g. `BUFFER_ATTR_UNITS="FT"`
- **BUFFER_COMBINE_AREAS** - boolean, whether to combine created buffers (**default** = NO). e.g. BUFFER_COMBINE_AREAS=**YES**
- **BUFFER_CREATE_RECTS** - boolean, whether to create rectangle buffers (default = NO). e.g. BUFFER_CREATE_RECTS="**YES**"
- **BUFFER_Z_OFFSET** - Z-offset applied to z-value of 3D features (**default** = **0**), in meters. e.g. `BUFFER_Z_OFFSET="100"`
- **BUFFER_ZONES** - number of buffer zones to create per feature (**default** = **1**). e.g. `BUFFER_ZONES=2`

Additional Vector Editing Options

- **DELETE_FEATURES** - specifies whether or not to mark all matching features as deleted. Use DELETE_FEATURES=**YES** to enable.
- **DELETE_ISLANDS** - use DELETE_ISLANDS=**YES** to specify that all islands/holes in matching area features should be marked as deleted
- **COORD_OFFSET** - specifies the offset to apply to any coordinates for the features that match the specified criteria. The offset should be in the units of the layer the features are being matched from. The offset should be specified as a *comma-delimited list of the X and Y offsets*, such as `COORD_OFFSET="100000.0,200000.0"`.
- **COORD_SCALE** - specifies the scale factors to apply to any coordinates for the features that match the specified criteria. Each coordinate will be multiplied by these scale factor. The scale factors should be specified either as a *single value* (the most common scenario) or as *separate scale factors for the X and Y values*, like `COORD_SCALE=1.00005` for a single value, or `COORD_SCALE=1.0045,1.0052` for separate X and Y coordinate scales.
- **ROTATE_ANGLE** - specifies the angle in degrees by which to rotate matching features by clockwise around some point. The ROTATE_ABOUT parameter specifies what point the feature coordinates are rotated about.
- **ROTATE_ABOUT** - if a non-zero ROTATE_ANGLE value is specified, specifies the point that the features should be rotated about, with the default being about the center of all matching features. The following values are supported:
    - **COMBINED_CENTER** - rotates about the combined center of all matching features. This is the default value.
    - **INDIVIDUAL_CENTER** - rotates each feature about the center of the feature independently.
    - *Manually specified location* - you can manually specify the X and Y coordinates to rotate about (in the projection of the layer being rotated), like `ROTATE_ABOUT="45000.0,1356000.0"`
- **CREATE_LABEL_POINTS** - specifies that a point feature should be created at the centroid/label position of each matching area feature. Use CREATE_LABEL_POINTS=**YES** to enable. If you only want to match areas with a non-empty label also add COMPARE_STR="<Feature Name>=*".
- **CREATE_VERTEX_POINTS** - specifies that a point feature should be created at each vertex of the matching area and line features. Use CREATE_VERTEX_POINTS=**YES** to enable.
- **CONNECT_ISLANDS** - specifies that any matching area features with islands should have the islands connected to the outer area boundary to form a single vertex list. This is useful if you need to export the data

for use in software that doesn't support area features with holes/islands in them. Use CONNECT_ ISLANDS=**YES** to enable.

- **SIMPLIFICATION** - specifies a simplification threshold to use to simplify/reduce the matching area and line features. It can contain a *horizontal threshold* (e.g., `SIMPLIFICATION="20"`) or *both horizontal and vertical thresholds, separated by a blank* (e.g., `SIMPLIFICATION="20 10"`).
    - The horizontal threshold specifies how far off a straight line (in the units of the current projection) that a point has to be before it is kept. Vertices that are this distance or further from the straight line will be kept. Vertices that are closer than this distance from the straight line will be discarded.
    - The vertical threshold applies only to 3D data, and will be applied to the vertex elevations. If the calculation exceeds the threshold, then the vertex will be kept.
    - By default, the vertical threshold is treated as elevation (in the vertical unit of the data). This specifies a threshold for the elevation difference between vertices. If the elevation difference is greater than or equal to the threshold, the vertex will be kept.
    - If you include the VERT_THRESH_IS_SLOPE parameter, the vertical threshold will be treated as slope, in degrees. This indicates a threshold for the slope difference between vertices. Each calculation involves three points (A, B, and C). If the difference between slope AC and slope AB exceeds the threshold, and the difference between slope AC and slope BC exceeds the threshold, then the point will be kept.

- **CONVERT_AREAS_TO_LINES** - specifies that any matching area features will have new line features created from them. Use CONVERT_AREAS_TO_LINES=**YES** to enable this functionality. By default each matching area and all islands/holes in that area will have lines created for them (the islands will get an ISLAND=Y attribute), but you can add IGNORE_ISLANDS=YES to just create lines from the parent areas.

- **INSERT_VERTICES_AT_INTERSECTIONS** - specifies that new vertices will be inserted at the intersection of any matching line features. Use INSERT_VERTICES_AT_INTERSECTIONS=**YES** to enable this functionality.

- **SPLIT_AT_INTERSECTIONS** - specifies that any matching line features that share an interior vertex will be split at that vertex into new lines. Use SPLIT_AT_INTERSECTIONS=**YES** to enable this functionality. Use this in conjunction with the INSERT_VERTICES_AT_INTERSECTIONS option to insert vertices at intersections and then split the lines there. The INSERT_VERTICES_AT_INTERSECTIONS option is enabled by default if SPLIT_AT_INTERSECTIONS is enabled (v16.1.2 and later). Add INSERT_VERTICES_AT_ INTERSECTIONS=**NO** if you only want to split at existing vertices.

- **PTS_AT_INTERSECTIONS** - specifies that new point features will be created wherever two or more matching line features touch. Use PTS_AT_INTERSECTIONS=**YES** to enable.

- **UNIQUE_ID_BASE** - specifies a number to start assigning unique IDs to features that match the query. For example, using `UNIQUE_ID_BASE=1` would assign ID attributes to each matching feature starting at 1 and increasing sequentially (i.e. 1, 2, 3, etc.).

- **SMOOTH_FEATURES** - specifies that matching area and line features should be smoothed. Use SMOOTH_ FEATURES=**YES** to enable.

- **CREATE_COVERAGE_AREAS** - specifies that a new layer with coverage areas for all matching features should be created. Use CREATE_COVERAGE_AREAS=**YES** to enable.

- **COVERAGE_SMOOTHING_FACTOR** - specifies a smoothing factor to use when creating the coverage areas to control how tightly shrink-wrapped around the vector features the area is. The **default** value is **1.0**, but any value *greater than 0* is allowed, with larger values resulting in more smoothing.

- **CREATE_SKELETON_LINES** - specifies that skeleton lines for the matching area features should be created and added to a new layer. The layer description will be the area layer description with "Skeleton Lines" appended to it. Use CREATE_SKELETON_LINES=**YES** to enable.

- **REMOVE_ISLANDS** - specifies that any areas with islands/holes should be chopped up to remove the islands. The original area will be marked as deleted and the new island-less areas added to the same layer. Use REMOVE_ISLANDS=**YES** to enable this behavior.

- **POINT_LAYER_TO_SPLIT_AGAINST** - specifies the full path and filename or description of a loaded point layer to use to split matching line features. The closest line to each point in the layer will be found and (if closer than MAX_DIST) will be split at the location that the point snaps to.

- **MAX_DIST** - specifies the maximum distance (in meters) that a point from the POINT_LAYER_TO_SPLIT_ AGAINST layer can be from the nearest line and still split the line.

- **FIX_INVALID** - specifies that invalid polygon geometries should be found and, if possible, fixed. Add FIX_INVALID=**YES** to enable this.

- **Specify Bounding Box for Operation**

    See also "Specify Bounds for Operation" on page 125

SAMPLES

Here is an example illustrating how to move features with a CLASS attribute with a value of '1' to a new layer named 'Major Highways':

```
EDIT_VECTOR MOVE_TO_NEW_LAYER=YES NEW_LAYER_NAME="Major Highways" \
  COMPARE_STR="CLASS=1"
```

Here is an example illustrating how to add evenly spaced points every 200 meters along the features containing a CLASS attribute with a value of '1'. Point features will also be created at the end point, and at each current vertex:

```
EDIT_VECTOR CREATE_POINTS_ALONG_FEATURES=200 COMPARE_STR="CLASS=1" \
  KEEP_END_POINT=YES KEEP_ORIGINAL_VERTICES=YES
```

Here is an example illustrating how to add evenly spaced perpendicular lines, 50 meters long, every 250 meters along the features containing a CLASS attribute with a value of '2'. Line features will also be created at the end point:

```
EDIT_VECTOR CREATE_PERP_LINES_ALONG_FEATURES=250 PERP_LINE_LENGTH=50 COMPARE_STR="CLASS=2" \
  KEEP_END_POINT=YES
```

Here is a sample script demonstrating applying elevations to all loaded vector data from all loaded terrain:

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
// Apply elevations from all loaded terrain layers to all loaded vector layer.
// Do NOT add elev values to existing values and do NOT include unit suffix.
// If the data already has an existing elevation, do NOT replace it
// Assign elevations for points to the ELEV_1 attribute rather than ELEVATION
EDIT_VECTOR APPLY_ELEVS=YES ADD_EXISTING_ELEV=NO INC_UNIT_SUFFIX=NO \
REPLACE_EXISTING=NO ELEV_ATTR="ELEV_1"
```

For more examples of how to use the EDIT_VECTOR command, see the sample at the bottom of this document.

COMBINE_LINES

The COMBINE_LINES command allows you to combine connected lines features based on one or more attribute or label values. You can choose to combines in just a single loaded layer or in all loaded vector layers. You can either create new line features from the connected lines, or using the CREATE_AREAS_FROM_LINES parameter instead create new area features by connecting the lines into closed shapes. The newly created features will be placed in a new layer and have the current projection. If creating lines, any lines that are connected to another line will be marked as deleted. You can also supply multiple COMPARE_STR parameters to apply multiple criteria, all of which must be true, in order for the lines to be considering for combining.

The following parameters are supported by the command:

- **CREATE_AREAS_FROM_LINES** - controls whether or not area features will be created from connected lines or just new line features. The **default** is CREATE_AREAS_FROM_LINES=NO. Use CREATE_AREAS_ FROM_LINES=**YES** to create new areas rather than lines.
- **FILENAME** - filename of the layer to assign types to. If an empty value is passed in, all loaded vector layers will be updated. This parameter can be listed more than once to specify multiple input files, like `FILENAME= E="FILENAME_1" FILENAME="FILENAME_2"`. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center.

- **COMPARE_STR** - specifies a comparison operation to perform to see if a line feature is one that needs to be considered. The format is *attr_name=attr_value* or *attr_name!=attr_value* (for equals and not equals comparisons respectively). For example if you have an attribute named CFCC and you want to match when the value of that attribute starts with an 'A', you can use COMPARE_STR="CFCC=A*" as your parameter. You can add multiple COMPARE_STR parameters to a single command to combine multiple criteria for your search. If you would like to compare against a feature label rather than an attribute, use **<Feature Name>** for your attribute name. To compare against a feature type name rather than an attribute, use **<Feature Type>** for your attribute name. To compare against the feature description, use **<Feature Desc>** for your attribute name. If you just want all lines features from the specified layer(s), just don't specify a COMPARE_STR value. You can also use **<Feature Layer Name>** to specify a match against a layer name that will support wildcards.

- **CASE_SENSITIVE** - specifies whether or not text comparisons are case sensitive or not. Use CASE_SENSITIVE=**YES** to enable, by default comparisons are not case sensitive.

- **LAYER_DESC** - specifies the name to assign to the newly generated layer containing the connected line features. If no layer description is provided, the default name of "Combined Lines" will be used.

- **COMPATIBLE_ONLY** - specifies whether or not any connecting lines should be combined (the default) or just those which have compatible types and attributes. Use COMPATIBLE_ONLY=**YES** to enable combining only compatible lines.

- **CLOSED_LINES_ONLY** - specifies whether or not any connected lines have to form a closed shape in order to create an area when generating area features. Use CLOSED_LINES_ONLY=**YES** to enable requiring closed paths.

- **CREATE_MULTIPLE_AREAS** - specifies whether multiple area features can be created if the lines don't all connect to each other. This is **enabled by default**, use CREATE_MULTIPLE_AREAS=**NO** to disable this and require all matching lines to connect in a single path before creating an area.

- **MAX_DIST** - specifies the maximum distance in meters that the end points of lines can be apart and still count as connected when combining them.

CROP_AREAS_TO_LINES

The CROP_AREAS_TO_LINES tool crops or splits areas based on line features. It uses the following parameters:

- **AREA_FILENAME** - filename of the layer that contains the area features to be cropped. If an empty value is passed in, all loaded area features that meet the AREA_COMPARE_STR conditions will be used. When running the script in the context of the main map view you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center.

- **AREA_COMPARE_STR** - specifies a comparison operation to perform to see if an area feature is one that needs to be cropped. The format is *attr_name=attr_value* or *attr_name!=attr_value* (for equals and not equals comparisons respectively). For example if you have an attribute named CFCC and you want to match when the value of that attribute starts with an 'A', you can use `COMPARE_STR="CFCC=A*"` as your parameter. You can add multiple AREA_COMPARE_STR parameters to a single command to combine multiple criteria for your search. If you would like to compare against a feature label rather than an attribute, use **<Feature Name>** for your attribute name. To compare against a feature type name rather than an attribute, use **<Feature Type>** for your attribute name. To compare against the feature description, use **<Feature Desc>** for your attribute name. If you just want all lines features from the specified layer(s), just don't specify a COMPARE_STR value. You can also use **<Feature Layer Name>** to specify a match against a layer name that will support wildcards.

- **LIST_FILENAME** - filename of the layer that contains the line features that will be used to crop the area features. If an *empty value* is passed in, all loaded line features that meet the LINE_COMPARE_STR conditions will be used. When running the script in the context of the main map view you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center.

- **LINE_COMPARE_STR** - specifies a comparison operation to perform to see if a line feature is one that will be used to crop the areas. The format is *attr_name=attr_value* or *attr_name!=attr_value* (for equals and not equals comparisons respectively). For example if you have an attribute named CFCC and you want to match when the value of that attribute starts with an 'A', you can use `COMPARE_STR="CFCC=A*"` as your parameter. You can add multiple AREA_COMPARE_STR parameters to a single command to combine multiple criteria for your search. If you would like to compare against a feature label rather than an attribute, use **<Feature Name>** for your attribute name. To compare against a feature type name rather than an

attribute, use **<Feature Type>** for your attribute name. To compare against the feature description, use **<Feature Desc>** for your attribute name. If you just want all lines features from the specified layer(s), just don't specify a COMPARE_STR value. You can also use **<Feature Layer Name>** to specify a match against a layer name that will support wildcards.

Attribute Management

ADD_MEASURE_ATTRS

The ADD_MEASURE_ATTRS command allows you to add/update feature measure attributes to all of the line and area features in a loaded vector layer.

The following parameters are supported by the command:

- **FILENAME** - filename of the layer to update. If an empty value is passed in, all layers that were created by the script, such as those from a GENERATE_CONTOURS command, will be updated. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center.

- **AREA_UNITS** - specifies the units to use when storing area measurements. The following values are supported:
    - **ACRES**
    - **HECTARES**
    - "**SQUARE FEET**"
    - "**SQUARE KILOMETERS**"
    - "**SQUARE METERS**"
    - "**SQUARE MILES**"

- **DISTANCE_UNITS**- specifies the units to use when storing linear distance measurements. The following values are supported:
    - **METRIC** - meters for shorter distances, kilometers for longer
    - **STATUTE** - feet for shorter distances, miles for longer
    - **YARDS** - yards for shorter distances, kilometers for longer
    - **NAUTICAL** - feet for shorter distances, nautical miles for longer
    - **CHAINS** - chains for shorter distances, miles for longer

- **MEASURE_UNIT_TYPE** - specifies how to handle measurement values of different sizes
    - **AUTO** - automatically use base units for smaller measurements and large units for long (i.e. meters for shorter distances, kilometers for longer)
    - **BASE** - always use base units, regardless of size. For example, always use meters for distance. Use this if you want to numerically compare values
    - **LARGE** - always use large units, regardless of size. For example, always use kilometers for distance. Use this if you want to numerically compare values

CALC_ATTR

The CALC_ATTR command allows you to calculate a new attribute value (or update the value for an existing attribute) for features in a layer based on a source attribute (including things like the feature label or type) and a

second value. The second value can be a specified string or number, or the value from another attribute of the feature.

The following parameters are supported by the command:

- **FILENAME** - filename of the layer to update. If an empty value is passed in, all loaded vector layers will be updated. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center If you don't pass anything in all vector layers will be operated on.
- **TYPE** - specifies what type of operation to use when assigning the new attribute value.
    - **COPY** - copies the source attribute value into the new attribute
    - **ADD** - numerically adds the second value to the source value and saves the result into the new attribute
    - **SUBTRACT** - numerically subtracts the second value from the source value and saves the result into the new attribute
    - **MULTIPLY** - numerically multiplies the second value by the source value and saves the result into the new attribute
    - **DIVIDE** - numerically divides the source value by the the second value and saves the result into the new attribute
    - **APPEND** - appends the second value (as a string) to the source value and saves the result into the new attribute. The SEP_STR parameter defined below is used to separate the second value from the source.
    - **PREPEND** - prepends the second value (as a string) to the source value and saves the result into the new attribute The SEP_STR parameter defined below is used to separate the second value from the source.
- **NEW_ATTR** - specifies the attribute value to create or update. See special [Attribute Name](#) parameter details.
- **SOURCE_ATTR** - specifies the attribute value to start with when creating the new attribute. See special [Attribute Name](#) parameter details.
- **VALUE_ATTR** - specifies the attribute value to use as the 2nd value of the calculation. See special [Attribute Name](#) parameter details.
- **VALUE** - specifies the value to use as the 2nd value of the calculation. For numeric operations this must be a number.
- **SEP_STR** - specifies the string to use to separate the source attribute and 2nd value when appending or prepending text. If not provided the **default** is no separator at all.

SAMPLE

Here is a sample of creating a new elevation attribute in feet from an elevation attribute (ELEV_M) in meters, including with an appended unit string.

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
// Create new ELEV_FT attribute with attribute in feet in any loaded layers
CALC_ATTR TYPE="MULTIPLY" NEW_ATTR="ELEV_FT" SOURCE_ATTR="ELEV_MT" VALUE="3.2808"
// Append the unit name to the new attribute
CALC_ATTR TYPE="APPEND" NEW_ATTR="ELEV_FT" SOURCE_ATTR="ELEV_FT" VALUE=" ft"
```

CALC_ATTR_FORMULA

The CALC_ATTR_FORMULA command allows you to calculate a new attribute value (or update the value for an existing attribute) for features in a layer, based on a formula that may contain numbers, strings or other attributes. A number of functions may be used as well.

The following parameters are supported by the command:

- **FILENAME** - filename of the layer to update. If an empty value is passed in, all loaded vector layers will be updated. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the

'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center If you don't pass anything in all vector layers will be operated on.

- **NEW_ATTR** - specifies the attribute value to create or update. See special [Attribute Name](#) parameter details.
- **CALC_MODE** - specifies the way that formula operations are performed when data types are ambiguous. Possible values are:
  - **AUTOMATIC** - calculations are performed as numeric if the first operand is numeric, or as string otherwise.
  - **NUMERIC** - calculations are performed as numeric.
  - **STRING** - calculations are performed as string.
- **FORMULA** specifies the formula to be used to create the new attribute values. See the [formula calculator reference](#) documentation for details.

SAMPLE

Here is a sample of creating a new elevation attribute in feet from an elevation attribute (ELEV_M) in meters, including with an appended unit string.

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
// Create new ELEV_FT attribute with attribute in feet in any loaded layers
CALC_ATTR_FORMULA NEW_ATTR="ELEV_FT" FORMULA="ELEV_MT * 3.2808"
// Append the unit name to the new attribute
 CALC_ATTR_FORMULA NEW_ATTR="ELEV_FT" FORMULA="ELEV_FT + ' ft'"
```

COPY_ATTRS

The COPY_ATTRS command copies one or more attributes from one layer of features to another. The attributes are copied spatially between feature types, like copying point attributes to the area features those points are contained in, etc. Not every combination of feature types is supported for copying. If you choose a combination that is not supported (like LINES to LINES) you will get an error message.

The following parameters are used by the COPY_ATTRS command:

- **LAYER1_FILENAME** - full path and filename of the loaded vector layer to copy attributes from. You can also pass in the full description of the loaded layer to use in case you want to use a layer not loaded from a file. Wildcards (i.e. '*' and '?') are supported in the filename, so to match all layers, use *****.
- **FROM_TYPE** - defines the type of features to copy attributes from. The following values are supported:
  - **AREAS** - copy attributes from areas
  - **LINES** - copy attributes from lines
  - **POINTS** - copy attributes from points
- **LAYER2_FILENAME** - full path and filename of the loaded vector layer to copy attributes to. You can also pass in the full description of the loaded layer to use in case you want to use a layer not loaded from a file. Wildcards (i.e. '*' and '?') are supported in the filename, so to match all layers, use *****.
- **TO_TYPE** - defines the type of features to copy attributes to. The following values are supported:
  - **AREAS** - copy attributes to areas
  - **LINES** - copy attributes to lines
  - **POINTS** - copy attributes to points
- **ATTR_TO_COPY** - specifies the name of an attribute to copy from the "from" features to the "to" features. You can include multiple instances of this parameter to copy multiple attributes. In addition to attribute names, you can specify <Feature Name> to copy the feature display label.
- **MAX_DIS**T - when copying from LINES to POINTS or POINTS to LINES, specifies the maximum distance that a point can be from the closest line and still have the attributes copied.
- **ENCLOSED_ONLY** - when copying attributes from AREAS to LINES, specifies whether or not the line features have to be completely inside the areas. Use ENCLOSED_ONLY=**YES** to specify that the lines must be completely inside the areas.
- **MULTI_AREA** - specifies how to handle assigning attributes from AREAS to POINTS or LINES when the point/line feature is inside multiple area features. The following values are supported:

- **NONE** - don't copy the attributes from any of the containing areas if there are more than one
  - **ALL** - copy the attributes of every area containing the point/line feature
  - **ALL_SORT** - copy the attributes of every area containing the point/line feature. If there are multiple copies of the same attribute (including existing values), separate the values with a comma and sort them alphabetically. So if there are values of an attribute 'South St', '83 Pine', and 'Main', the result would be '83 Pine,Main,South St'. The sorting is done without regard to case.
  - **FIRST** - only copy the attributes of the topmost containing area to the point/line feature
- **MULTI_POINT** - specifies how to handle assigning attributes from POINTS to AREAS or LINES when there are multiple points inside a single area or near a line. The following values are supported:
  - **NONE** - don't copy the attributes from any of the points in the area/line feature
  - **ALL** - copy the attributes of every point in the area or near the line to the area/line feature
  - **ALL_CLONE** - if multiple exist clone the feature for each matching feature
  - **FIRST** - (to AREAS only) only copy the attributes of the first point in the area to the area
  - **CLOSEST** - (to LINES only) only copy the attributes of the closest point to the line to the line
  - **MIN** - (to LINES only) use the minimum value of the attribute encountered
  - **MAX** - (to LINES only) use the maximum value of the attribute encountered
- **AREA_COVERAGE** - specifies how covered a "to" area has to be by a "from" area for the attributes to be copied. The following values are supported:
  - **COMPLETE** - the "to" area has to be complete inside the "from" area
  - **PARTIAL** - the "to" area has to be covered at least some by the "from" area
  - **CENTROID** - the "to" area centroid has to be inside the "from" area. This option is the fastest but may result in some areas getting attributes that aren't covered at all if the centroid is outside of the "to" area.
  - **MOST** - the "to" area has to be at least half covered by the "from" area
- **EQUAL_ATTR** - specifies an attribute that must be equal between two features in order to be a possible match for copying other attributes. See special [Attribute Name](#) parameter details for allowed attribute names. You can include multiple EQUAL_ATTR parameters on a command to require multiple attributes be equal.

SAMPLE

```
GLOBAL_MAPPER_SCRIPT VERSION="1.00"
/* Copy all loaded area attributes to the points in the area */
COPY_ATTRS LAYER1_FILENAME="*" FROM_TYPE="AREAS" LAYER2_FILENAME="*" TO_TYPE="POINTS"
```

GENERATE_REPORT

The GENERATE_REPORT command allows you to generate a CSV text report file on the data in one or more loaded layers broken down by a particular attribute value, feature name, or type, or just a single line report about all features. The report will include the count of area, line, and point features matching the specified criteria as well as the total combined length of the line features and combined covered area of the area features.

The following parameters are supported by the command:

- **OUTPUT_FILENAME** - specifies the name of the text .csv file to write the report results to.
- **FILENAME** - filename of the layer to generate the report for. If an empty value is passed in, all layers that were created by the script, such as those from a GENERATE_CONTOURS command, will be used to generate the report. This parameter can be listed more than once to specify multiple input files, like `FILENAME-E="FILENAME_1" FILENAME="FILENAME_2"`. You can also pass in the value **'USER CREATED FEATURES'** when running a script in the context of the main map view or loading a workspace to have the 'User Created Features' layer be used. If no FILENAME parameter is provided, the report will cover all available vector layers.
- **REPORT_ATTR** - specifies what to use to divide up the report into categories. See special [Attribute Name](#) parameter details. If no REPORT_ATTR parameter is provided or the value is empty, only a single line concerning all matching features will be generated in the report. You can provide multiple REPORT_ATTR parameters if you want to group the results by more than one attribute. For example you might want to separate by layer and type so you could add `REPORT_ATTR="<Feature Layer Name>"` and `REPORT_ATTR="<Feature Name>"`.
- **COMPARE_STR** - specifies a comparison operation to perform to see if a feature is one that needs to be included in the report. The format is *attr_name=attr_value*. For example if you have an attribute named CFCC and you want to match when the value of that attribute starts with an 'A', you can use `COMPARE_`

`STR="CFCC=A*"` as your parameter. You can add multiple COMPARE_STR parameters to a single command to combine multiple criteria for your search. See special Attribute Name parameter details for special attribute names to compare against.

- **CASE_SENSITIVE** - specifies whether or not text comparisons are case sensitive or not. Use CASE_ SENSITIVE=**YES** to enable, by **default** comparisons are not case sensitive.

JOIN_TABLE

The JOIN_TABLE command joins the attributes from a table file to the features of a loaded vector layer. The following parameters are supported by the command:

- **LAYER_NAME** - vector layer to join the attributes to.
- **FILENAME** - file that contains the attributes to join to the vector layer features.
- **JOIN_FILE_ATTR_NAME** - name of the file attribute to join on.
- **JOIN_LAYER_ATTR_NAME** - name of the layer attribute to join on.
- **FILE_DELIM** - string that delimits the values in a text file. You can also use FILE_DELIM=**SPACE** for a space or FILE_DELIM=**TAB** for a tab delimiter.
- **CASE_SENSITIVE** - specifies whether or not text comparisons are case sensitive or not. Use CASE_ SENSITIVE=**YES** to enable, by default comparisons are not case sensitive.
- **IGNORE_WHITESPACE** - specifies whether or not whitespace (i.e. spaces and tabs) should be ignored when looking for matches to join on. Use IGNORE_WHITESPACE=**YES** to ignore whitespace, by default whitespace is considered.
- **ALLOW_DUPLICATES** - specifies how duplicate entries in the join file should be handled. By default the last matching entry will be used, but you can set the behavior as follows:
    - **KEEP_LAST** (or **YES**) - keep the attribute values from the last matching record in the join file
    - **KEEP_FIRST** - keep the attribute values from the first matching record in the join file
    - **KEEP_ALL_APPEND** - keep all matching records from the join file, appending new values to the existing attribute with a comma separator
    - **KEEP_ALL_APPEND_SORT_ASC** - same as KEEP_ALL_APPEND, except that the list of values will be sorted in ascending order. If all of the values are numbers, they will be sorted using a numeric sort, otherwise, they will be sorted using a text sort.
    - **KEEP_ALL_APPEND_SORT_DESC** - same as KEEP_ALL_APPEND, except that the list of values will be sorted in descending order. If all of the values are numbers, they will be sorted using a numeric sort, otherwise, they will be sorted using a text sort.
    - **KEEP_ALL_COPY** - keep all matching reords from the join file, but create new attribute values with a numeric suffix for the multiple entries. For example if there were 5 matching records with an ADDR attribute, you would get ADDR, ADDR2, ADDR3, ADDR4, and ADDR5 attributes added
    - **COPY_FEATURE** - create duplicate features for each record with a duplicate join attribute, one addition feature for each duplicate join attribute value
    - **PROMPT** - prompt the user for how to handle duplicates if encountered
    - **NO** - abort the join if any duplicate records are encountered in the join file

Attribute Name Values

An attribute name parameter has a value that can be either the name of an attribute or one of the special values below for accessing other fields associated with the feature:

- **<Feature Name>** - the value is the display label of the feature
- **<Feature Type>** - the value is the classification of the feature
- **<Feature Desc>** - the value is the description of the feature (often the same as the feature type). For formats like DXF or DGN, this will be the DXF layer or DGN level name respectively.
- **<Feature Layer Name>** - the value is the name of the layer that the feature is in
- **<Feature Layer Group Name>** - the value is the name of the layer group that the feature is in
- **<Feature Source Filename>** - the value is the filename of the file the layer was loaded from
- **<Index in Layer>** - the value is the 0-based index of the feature in the layer that the feature is in. Note that area, line, and point features are separately indexed.

Raster Analysis

APPLY_FORMULA

The APPLY_FORMULA command allows you to apply a mathematical formula to the bands in one or more loaded raster layers to create a new raster or grid layer. This is useful for doing things like NDVI or NDWI calculations from multi-spectral imagery, among a large array of other multi-spectral analysis.

The following parameters are supported by the command:

- **FILENAME** - filename of the layer(s) to use as input to the operation. You can use * to use all loaded raster imagery layers. This is the default. This parameter can be listed more than once to specify multiple input files, like `FILENAME="FILENAME_1" FILENAME="FILENAME_2"`. You can also pass in the description of the layer if it isn't based on a file, such as a layer created by the script. When running the script in the context of the main map view (including loading a workspace) you can also pass in the value '**USER CREATED FEATURES**' to have the 'User Created Features' layer updated or '**SELECTED LAYERS**' to have any layers selected in the Control Center

- **LAYER_DESC** - description to assign to the newly created layer

- **FORMULA** - this free-form string describes *the formula for a single band of the output.* You would have a separate FORMULA parameter for each output band. So if you were creating a 3-band output you would have 3 separate FORMULA parameters. Each FORMULA parameter is a string that can either be the name of a pre-defined operation, like **NDVI** or **NDWI**, or a free-form equation definition. See formula reference documentation for details. The bands default to the original source bands from the provided input layers, but you can use the NUM_BANDS, BAND_BIT_DEPTH, and BAND_EXPORT_SETUP parameters (described below) to customize which bands from which input layers are assigned to each band number variable in the formula.

- **OUTPUT_BIT_DEPTH** - specifies the bit depth for the output band(s) (one for each FORMULA parameter). Valid values are **8**, **16**, or **32**. If you don't specify this then you will get 32-bit floating point values if the output is a single band grid and 8-bit per band for imagery output.

- **OUTPUT_GRID** - specifies whether the output should be treated as a grid that is shaded using an "elevation" shader or as an imagery layer. Use OUTPUT_GRID=**YES** to make it a shaded grid.

- **NUM_BANDS** - specifies how many bands of data to make available for input. If you don't specify a value for this the band count will be the maximum available for any of the loaded layers.

- **BAND_BIT_DEPTH** - specifies how many bits to use for each band of data. If you don't specify a value for this the highest bit depth of any of the input data layers will be used. The valid values are BAND_BIT_DEPTH=**8**, BAND_BIT_DEPTH=**16**, or BAND_BIT_DEPTH=**32**.

- **BAND_EXPORT_SETUP** - allows you to override the default band assignment for the FORMULA parameters. Use the following format to specify what band from what layer to use for a given export band: *<output_band>?<input_band>?<layer_filename>* . So for example to assign the 4th (infrared) band in an export from the 1st (red) band in a previously loaded file name C:\data\input_file.tif, use the following parameter: `BAND_EXPORT_SETUP="4?1?c:\data\input_file.tif"`. Note that you would include a separate BAND_EXPORT_SETUP parameter for each output band that you want to setup. If you leave off the filename then you all loaded data will be considered as input, with just the input-to-output band assignment being updated.

SAMPLE

Here is an example script command showing how to apply a formula to all currently loaded raster layers.

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
APPLY_FORMULA FILENAME="*" OUTPUT_BIT_DEPTH="8" FORMULA="IF(B1<90, B1*0.765+20, B1)"
```

PAN_SHARPEN

The PAN_SHARPEN command fuses a lower resolution color/multi-band image with a higher resolution pan-chromatic (grayscale) image to create a new color/multi-band image at the same detail as the pan image. This is often used with satellite-based imagery with a pan sensor at double the resolution as the color/multi-spectral sensor. The result is a new color/multi-band layer. In addition, all of the option parameters for the IMPORT command are also supported for this command.

The following parameters are used by the PAN_SHARPEN command:

- **COLOR_LAYER** - full path and filename of the color/multi-band layer to pan sharpen. You can also pass in the full description of the loaded layer to use in case you want to use a layer not loaded from a file.

- **PAN_LAYER** - full path and filename of the panchromatic layer to use. You can also pass in the full description of the loaded layer to use in case you want to use a layer not loaded from a file.

- **LAYER_DESC** - description to use for the new layer. If not provided, the description will be that of the color layer with (Pan Sharpened) appended.

- **ALGORITHM** - specifies which pan sharpening algorithm will be used to perform the operation. The following values are supported:
    - **IHS** - converts the color to the HSI (hue/saturation/intensity) colorspace, then replaces the intensity with the pan value. This is the default value.
    - **BROVEY** - uses the Brovey algorithm
    - **ESRI** - uses the Ersi pan sharpening transformation
    - **MEAN** - uses a simple mean (average) of the pan value and each band for the new band value

- **WEIGHTS** - provides a list of weight values to apply to each band in the color image. This should be a *comma-delimited list of 3 (RGB) or 4 (RGBI) weights* to apply. The weight order is *red, green, blue, infrared*. If not provided, each band will be equally weighted when calculating an initial intensity for the IHS and BROVEY algorithms. The ESRI algorithm uses a default weighting of "0.167,0.167,0.167,0.5" (i.e. half to infrared and equally divided amongst the rest). The weight values *will automatically be normalized* so they sum up to 1 for all used bands, so if you want to weight green at twice the other bands, you could use `WEIGHTS="1,2,1,1"`.

- **SPATIAL_RES** - specifies spatial resolution. Defaults to the minimum spatial resolution of the two layers if not specified. Should be formatted as *x_resolution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to export at 30 meter spacing, the parameter/ value pair would look like `SPATIAL_RES=30.0,30.0`. You can also specify as a percentage of the default resolution by adding a percent. For example to get half the detail your double the spatial resolution value, so you would use `SPATIAL_RES="200%,200%"`.

- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will automatically be converted to the curent view/export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.

- **Specify Bounding Box for Operation**

    See also "Specify Bounds for Operation" on page 125

SAMPLE

Example: Create a pan sharpened layer from 2 loaded 16-bit images:

```
// Define variables with filename for color and pan layers
DEFINE_VAR NAME="COLOR_FNAME" VALUE="LC80330322015307LGN00_B4325_RGBI"
DEFINE_VAR NAME="PAN_FNAME" VALUE="LC80330322015307LGN00_B8_PAN.TIF"
// Load color and pan layers
```

```
IMPORT FILENAME="%COLOR_FNAME%" TRANSPARENT_COLOR="RGB(255,255,255)" \
 AUTO_CONTRAST="YES" CONTRAST_SHARED="NO" CONTRAST_STRETCH_SIZE="2.000" CONTRAST_
MODE="PERCENTAGE"
IMPORT FILENAME="%PAN_FNAME%" LOAD_FLAGS="0~0~0~1~0~0" TRANSPARENT_COLOR="RGB(0,0,0)" \
 AUTO_CONTRAST="YES" CONTRAST_SHARED="NO" CONTRAST_STRETCH_SIZE="0.000" CONTRAST_
MODE="PERCENTAGE"
// Pan Sharpen images
PAN_SHARPEN COLOR_LAYER="%COLOR_FNAME%" PAN_LAYER="%PAN_FNAME%" ALGORITHM="BROVEY"
```

GENERATE_EQUAL_VAL_AREAS

The GENERATE_EQUAL_VAL_AREAS command allows for the generation of areas for regions of the same (or similar) color, elevation, or slope values from a loaded raster or elevation layer.

- **FILENAME** - specifies the filename of the already loaded layer from which to generate the equal-value areas. This parameter can be listed more than once to specify multiple input files, like `FILENAMEE="FILENAME_1" FILENAME="FILENAME_2"`.

- **LAYER_DESC** - specifies the description to use for the layer

- **ATTR_NAME** - specifies the name to use for the attribute holding the color, elevation, or slope that a particular area represents. If you leave this attribute off then no value attribute will be saved with each area.

- **ATTR_FORMAT** - specifies a custom format string for creating the color attribute. This would be a C-style string with either a single number parameter or 3 for separate red, green, and blue. Usually you wouldn't specify this and the default would then just be the same as ATTR_FORMAT="RGB(%3d, %3d, %3d)".

- **AREA_TYPE** - specifies the name of the area type to assign to the area features. See the Area Styles tab of the Configuration dialog for a list of available type names.

- **EQUAL_COLORS** (raster only) - specifies the color(s) to create areas for. If you don't provide one of more colors all colors are matched on. The color(s) should be specified as *RGB(<red>,<green>,<blue>)*. For example, to create areas for white, use `EQUAL_COLORS=RGB(255,255,255)`. Optionally, if the image that you are creating areas for uses a palette for the colors, you can specify a palette index in the following format: *INDEX(<0-based palette index>)*. For example, to make the second color in the palette transparent, use `EQUAL_COLORS=INDEX(1)`. You can provide more than one color to match on by providing a semi-color separated list, like `EQUAL_COLORS="RGB(0,255,0);RGB(255,0,0)"` to match on blue and red.

- **COLOR_DIST** - specifies how far from an exact match each color channel of a color value must be to be considered the same. By default a value of **0** is used which means exact matches only. If you want to break the entire color range into say 4 ranges for each color channel, use something like `COLOR_DIST=32` as that would allow colors up to 32 away from each color channel value to match to a color.

- **ELEV_DIST** - specifies how far from an exact match (in meters) each value must be to be considered the same. By default a value of **0** is used which means exact matches only. If you want to say split into area groups 10 meters in size, use `ELEV_DIST=5`. This would give you areas with values between -5 and 5 meters, 5 and 15 meters, etc.

- **SLOPE_DIST** - specifies how far from an exact match (in degrees) each value must be to be considered the same. By default a value of **0** is used which means exact matches only. If you want to say split into area groups 10 degrees in size, use `SLOPE_DIST=5`. This would give you areas with values between 0 and 10 degrees, 10 and 20 degrees, etc.

- **FORCE_RGB** - specifies that the attribute value for a color-based area will always be the full RGB color and not a palette index/name if available. Use FORCE_RGB=**YES** to enable.

- **FIX_INVALID** - specifies whether or not the resulting areas should be split up into pieces if any invalid (i.e. self-intersecting) pieces are created. Use FIX_INVALID=**YES** to force the invalid areas to be fixed.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

SAMPLE

This sample will generate equal-elevation areas of size 20 meters (10 meters on either side) from the specified DEM layer and store the elevation values in an attribute named ELEV for each area feature.

```
  GENERATE_EQUAL_VAL_AREAS FILENAME="C:\temp\export test\blue_springs_4_quads.dem" ELEV_DIST=10.0
ATTR_NAME="ELEV"
```

Export

EXPORT_ANY

The EXPORT_ANY command exports loaded data to some file format. The TYPE parameter for this will automatically determine which of the EXPORT_ELEVATION, EXPORT_PACKAGE, EXPORT_PDF, EXPORT_RASTER, or EXPORT_ VECTOR commands should be used. For example if you use TYPE=USGS_DEM then you will export using the EXPORT_ELEVATION command. The parameters for the command that is used based on the TYPE parameter.

- **TYPE** - type of file we're exporting to. See the other EXPORT commands for a complete list of available types. The following types are special types:
  - **FIRST_LOADED** - export to the same type as the first loaded file
  - **LAST_LOADED** - export to the same type as the last loaded file

EXPORT_CLOUD

The EXPORT_CLOUD command exports data to a cloud database, including Amazon S3 account. The following parameters are supported by the command.

- **CLOUD_TYPE** - Cloud type, currently only AWS is supported "**Amazon's AWS S3**"
- **CLOUD_KEY1**- first access key, for AWS S3 this is the Public Key
- **CLOUD_KEY2** - second access key, for AWS S3 this is the Private Key
- **CLOUD_FOLDER** - folder where file exists in the cloud, for AWS S3 this is the bucket
- **CLOUD_LOCATION** - location where folder exists, for AWS S3 this is region
- **CLOUD_FILE** - the name of the file as it exists in the cloud.

SAMPLE

```
EXPORT_CLOUD GEN_PRJ_FILE=YES TYPE="SHAPEFILE" \
ELEV_UNITS="METERS" LABEL_FIELD_FORCE_OVERWRITE="YES" LABEL_FIELD_SEP="0x20" LABEL_FIELD="NAME"
\
CODE_PAGE="0" CLOUD_TYPE="Amazon's AWS S3"  CLOUD_KEY1="AAAAAAAAAAAAAAAAAAAA" CLOUD_
KEY2="AAAA+bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb" \
CLOUD_FOLDER="bmg-bucket" CLOUD_LOCATION="us-east-1" CLOUD_FILE="biggybig.shp"
```

EXPORT_ELEVATION

The EXPORT_ELEVATION command exports all currently loaded elevation data to a file. The following parameters are supported by the command.

- **FILENAME** - full path to file to save the data to

- **EXPORT_LAYER** - filename or description of layer(s) to export. By default all compatible and exportable layers are exported. You can include multiple EXPORT_LAYER parameters if you have multiple masks to search. Wildcards (* and ?) are supported. Hidden layers are not considered. To export just the layer(s) selected in the Control Center, use EXPORT_LAYER="**SELECTED LAYERS**".

- **TYPE**

  - type of elevation file we're exporting to:

  - **FIRST_LOADED** - export to the same type as the first loaded file
  - **LAST_LOADED** - export to the same type as the last loaded file
  - **ARCASCIIGRID** - export an Arc ASCII Grid format file.
  - **BIL** - export to a BIL format file (simple binary format with header).
  - **BT** - export to a BT (Binary Terrain) format grid file.
  - **DTED** - export to DTED format grid files.
  - **DXF_3DFACE** - export a DXF 3D face format file.
  - **DXF_MESH** - export a 3D DXF mesh format file.
  - **DXF_POINT** - export a 3D DXF point format file.
  - **ERDAS** - export to an Erdas Imagine IMG format file.
  - **FLOATGRID** - export a Float/Grid format file.
  - **GEOSOFT_GRID** - export to a Geosoft grid format file.
  - **GEOTIFF** - export to a GeoTIFF format file.
  - **GLOBAL_MAPPER_GRID** - export to a Global Mapper Grid format file.
  - **GRAVSOFT_GRID** - export to a Gravsoft Grid format file.
  - **GWS** - export to Windsim GWS.
  - **HF2** - export to a HF2/HFZ format file.
  - **IDRISI_RASTER** - export to an Idrisi elevation format file.
  - **LEVELLER_HF** - export to a Leveller heightfield file.
  - **LIDAR_LAS** - export to a Lidar LAS/LAZ file. Use .laz in filename to get LasZip.
  - **MAPMAKERTERRAIN** - export to a MapMaker Terrain format file.
  - **NITF** - NITF format terrain file
  - **PLS_CADD** - PLS-CADD terrain file
  - **ROCKWORKS_GRID** - export to a RockWorks Grid format file.

  - **SRTM** - exports SRTM HGT 1x1 degree tiles. See SRTM_LEVEL in Other Format Specific fields to set the level.

  - **STL** - export to a STL format file
  - **SURFERGRID** - export to a Surfer Grid format file. The FORMAT parameter specifies whether it is an ASCII or binary format Surfer Grid file.
  - **TERRAGEN** - export to a Terragen terrain file.
  - **USGS_DEM** - export to a native format USGS DEM file.
  - **VRML** - export to a VRML file.
  - **VULCAN_3D** - export to a Vulcan3D triangulation file.
  - **XYZ_GRID** - export to a XYZ Grid file.

- **ELEV_UNITS** - specify elevation units to use in export

  - **FEET** - export in US feet
  - **DECIFEET** - export in 10ths of US feet
  - **METERS** - export in meters
  - **DECIMETERS** - export in 10ths of meters
  - **CENTIMETERS** - export in centimeters

- **SPATIAL_RES** - specifies spatial resolution. Defaults to the minimum spatial resolution of the two layers if not specified. Should be formatted as *x_resolution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to export at 30 meter spacing, the parameter/value pair would look like `SPATIAL_RES=30.0,30.0`. You can also specify as a percentage of the default resolution by adding a percent. For example to get half the detail your double the spatial resolution value, so you would use `SPATIAL_RES="200%,200%"`.

- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will automatically be converted to the curent view/export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.

- **FORCE_SQUARE_PIXELS** - if this value is set to YES, the spatial resolution of the resultant elevation file will be set so that the x and y pixel size are the same, with the minimum default size being used for both.
- **SAMPLING_METHOD** (elevation and raster only) - specifies the sampling method to use when resampling this layer. The following values are supported:
  - **DEFAULT** - Use either automatic resampling based on export or layer resampling, depending on setting of global flag about whether to resample on export
  - **AUTO** - Automatically select a resampling method based on how the export resolution and bounds compare to the original layout for a layer. For example if you export to a lower resolution a box averager of appropriate size may be used automatically
  - **LAYER** - Use the sampling method set for each layer
  - The list of SAMPLING_METHOD values for the IMPORT command can also be specified to use a particular sampling method for all layers being exported.
    - **Shared IMPORT SAMPLING_METHOD values**

      See "SAMPLING_METHOD (elevation and raster only) - specifies the sampling method to use when resampling this layer. " on page 39
- **GEN_WORLD_FILE** - specifies that a world file (TFW for TIF, JGW for JPG, etc.) should be generated in addition to the grid file. Use GEN_WORLD_FILE=**YES** to turn on.
- **GEN_TAB_FILE** - specifies that a MapInfo TAB file should be generated in addition to the grid file. Use GEN_TAB_FILE=**YES** to turn on.
- **GEN_PRJ_FILE** - specifies that a projection (PRJ) file should be generated in addition to the data file. Use GEN_PRJ_FILE=**YES** to turn on.
- **GEN_AUX_XML_FILE** - specifies that an ESRI .aux.xml projection file should be generated in addition to the data file. Use GEN_AUX_XML_FILE=**YES** to turn on.
- **OVERWRITE_EXISTING** - specifies that existing files should be overwritten. The **default** is OVERWRITE_EXISTING=**YES**, so use OVERWRITE_EXISTING=**NO** to skip exporting files that already exist.
- **FILL_GAPS** - specifies that small gaps in between and within the data sets being exported will be filled in by interpolating the surrounding data to come up with an elevation for the point in question. This option is **on by default**, specify FILL_GAPS=**NO** to turn off.

BIL Grid Fields

- **BYTES_PER_SAMPLE** - specifies how many bytes to use per elevation sample in the BIL file. Valid values are **1** byte (8 bits), **2** bytes (16-bits) and **4** bytes (32-bits). If this value isn't specified, **2** bytes are used.
- **USE_UNSIGNED** - if exporting to 1 or 2 byte integer samples, add USE_UNSIGNED=**YES** to the command to get unsigned 8-bit or 16-bit values rather than the default signed values.
- **USE_BIG_ENDIAN** - specifies that elevation values written to BIL files should use big endian (Motorola byte order) rather than the default little endian (Intel byte order).
- **USE_ESRI_HDR** - specifies that the ESRI .hdr format should be used for the export. Use USE_ESRI_HDR=**YES** to enable.

XYZ Grid Fields

- **COORD_DELIM** - specifies the delimeter between coordinates
  - **COMMA** - coordinates are separated by commas
  - **FIXED_WIDTH** - coordinates are stored in fixed width columns
  - **SEMICOLON** - coordinates are separated by semicolons
  - **SPACE** - coordinates are separated by space characters
  - **TAB** - coordinates are separated by tab characters
- **EXPORT_COLORS** - specifies that RGB color values should be exported for each coordinate that is saved.
- **EXPORT_ECEF** - specifies that the export should use ECEF (earth-centered earth-fixed, or geocentric) coordinates.
- **EXPORT_ROW_MAJOR** - specifies that the XYZ values should be exported in row-major order. Use EXPORT_ROW_MAJOR=**YES** to enable this.
- **REVERSE_ROWS** - specifies that the values should be exported from bottom-to-top rather than top-to-bottom
- **REVERSE_COLS** - specifies that the values should be exported from right-to-left rather than left-to-right
- **THIN_GRID_MAX_DELTA** - specifies that the output points should be thinned in areas with elevation values that change less than the specified valu. The units are whatever units are specifed in the ELEV_UNITS parameter. By default if this value is provided the data will be divided in to grids of a max size 64x64. If all values in that grid are within THIN_GRID_MAX_DELTA of each other, only a single point will be exported. If

not, 4 smaller grids of size 32x32 are examined, and so on until a single point is reached. Use THIN_GRID_
PIX_SIZE to specify a different maximum subgrid size other than 64x64.

- **THIN_GRID_PIX_SIZE** - specifies the maximum number of samples on each side of the sub-grids examined with THIN_GRID_MAX_DELTA is specified.

- **COORD_DELIM** - specifies the delimeter between coordinates
    - **COMMA** - coordinates are separated by commas
    - **FIXED_WIDTH** - coordinates are stored in fixed width columns
    - **SEMICOLON** - coordinates are separated by semicolons
    - **SPACE** - coordinates are separated by space characters
    - **TAB** - coordinates are separated by tab characters

- **EXPORT_COLORS** - specifies that RGB color values should be exported for each coordinate that is saved.

- **EXPORT_ECEF** - specifies that the export should use ECEF (earth-centered earth-fixed, or geocentric) coordinates.

- **EXPORT_ROW_MAJOR** - specifies that the XYZ values should be exported in row-major order. Use EXPORT_ROW_MAJOR=**YES** to enable this.

- **REVERSE_ROWS** - specifies that the values should be exported from bottom-to-top rather than top-to-bottom

- **REVERSE_COLS** - specifies that the values should be exported from right-to-left rather than left-to-right

- **THIN_GRID_MAX_DELTA** - specifies that the output points should be thinned in areas with elevation values that change less than the specified valu. The units are whatever units are specifed in the ELEV_UNITS parameter. By default if this value is provided the data will be divided in to grids of a max size 64x64. If all values in that grid are within THIN_GRID_MAX_DELTA of each other, only a single point will be exported. If not, 4 smaller grids of size 32x32 are examined, and so on until a single point is reached. Use THIN_GRID_
PIX_SIZE to specify a different maximum subgrid size other than 64x64.

- **THIN_GRID_PIX_SIZE** - specifies the maximum number of samples on each side of the sub-grids examined with THIN_GRID_MAX_DELTA is specified.

ERDAS Fields

- **BYTES_PER_SAMPLE** - specifies how many bytes to use per elevation sample in the IMG file. Valid values are **2** bytes (16-bits) and **4** bytes (32-bits). If this value isn't specified, 2 bytes are used.

- **ADD_OVERVIEW_LAYERS** - specifies that overview (pyramid) layers should be generated for the export. Use ADD_OVERVIEW_LAYERS=**YES** to enable.

- **BLOCK_SIZE** - specifies the block size to use for the export. The default is BLOCK_SIZE=64.

- **COMPRESS_OUTPUT** - specifies whether or not the exported file should be compress. The default is COMPRESS_OUTPUT=**YES**.

- **BYTES_PER_SAMPLE** - specifies how many bytes to use per elevation sample in the IMG file. Valid values are **2** bytes (16-bits) and **4** bytes (32-bits). If this value isn't specified, 2 bytes are used.

- **ADD_OVERVIEW_LAYERS** - specifies that overview (pyramid) layers should be generated for the export. Use ADD_OVERVIEW_LAYERS=**YES** to enable.

- **BLOCK_SIZE** - specifies the block size to use for the export. The default is BLOCK_SIZE=64.

- **COMPRESS_OUTPUT** - specifies whether or not the exported file should be compress. The default is COMPRESS_OUTPUT=**YES**.

GeoTIFF Fields

- **BYTES_PER_SAMPLE** - specifies how many bytes to use per elevation sample in the vertical GeoTIFF file. Valid values are **2** bytes (16-bits) and **4** bytes (32-bits). If this value isn't specified, 2 bytes are used.

- **USE_UNSIGNED** - if exporting 2 byte integer samples, add USE_UNSIGNED=**YES** to the command to get unsigned 16-bit values rather than the default signed values.

- **TILE_SIZE** - specifies that the GeoTIFF file should be exported with a tiled organization and use the specified tile size. This tile size should be between **8** *and* **2048**. Typical values are **64**, **128**, and **256**.

- **DISABLE_BIGTIFF** - use to disable the automatic creation of BigTIFF-format files for very large exports. Use DISABLE_BIGTIFF=**YES** to disable the automatic BigTIFF support.

- **TIFF_COPYRIGHT** - specify text to store in TIFFTAG_COPYRIGHT tag.

- **TIFF_DATETIME** - specify text to store in TIFFTAG_DATETIME tag.

- **TIFF_DOC_NAME** - specify text to store in TIFFTAG_DOCUMENTNAME tag.
- **TIFF_GT_CITATION** - specify text to store in GeoTIFF GTCitationGeoKey GeoTIFF tag.
- **TIFF_IMAGE_DESC** - specify text to store in TIFFTAG_IMAGEDESCRIPTION tag.
- **TIFF_PCS_CITATION** - specify text to store in GeoTIFF PCSCitationGeoKey GeoTIFF tag.
- **TIFF_NO_GTIFF_HEADER** - don't embed a GeoTIFF header in the file. Use TIFF_NO_GTIFF_HEADER-R=YES to disable write of header.
- **COMPRESSION** (GeoTIFF only)- specifies the type of compression to use for the generated TIFF file. If you do not provide a compression value then no compression will be used.
  - **NONE** - Do not compress the output.
  - **LZW** - Use LZW (lossless) compression on the output.
  - **DEFLATE** - Use Deflate/ZIP (lossless) compression on the output.
- **VERT_CITATION** - specifies the text description to store in the GeoTIFF file for the vertical coordinate system for the elevations. If nothing is supplied the default one (if any) for the supplied VERT_CS_CODE will be used.
- **VERT_CS_CODE** - specifies the vertical coordinate system (i.e. vertical datum) to store in the GeoTIFF file to specify what the elevations are referenced to. Use the *EPSG code*, like **5103** for NAVD88. If you don't specify a value and you specify a VERT_CITATION field that matches a known name, the associated VERT_CS_CODE for that name will be used. For example, VERT_CITATION="NAVD88" will provide VERT_CS_CODE-E=5103 by default. If neither of these are present and the source files used all use the same known system, that will be used. Note that no vertical datum conversion is done, this is just to supply metadata.

FLOAT_GRID Fields

- **EXPORT_SLOPE** - use to specify that slope values should be exported rather than elevation values. Use EXPORT_SLOPE=**YES** to enable.
- **EXPORT_SLOPE_DIR** - use to specify that slope direction values should be exported rather than elevation values. Use EXPORT_SLOPE_DIR=**YES** to enable.
- **EXPORT_SLOPE_PERCENT** - use to specify that slope values should be exported as percent slope rather than degrees. Use EXPORT_SLOPE_PERCENT=**YES** to enable and also make sure to add EXPORT_SLOPE=**YES** to the command line.

DTED Fields

- **DTED_LEVEL** - specifies which DTED level to export to. The values must be between **0** and **5** (with **0**, **1**, and **2**) being the only levels supported by most applications.
- **DTED_PRODUCER_CODE** - specifies a producer code to store in the DTED file header. This should be in the form CCAAABBB where the first 2 characters are the country code.
- **DTED_SECURITY_CLASS** - specifies the security classification character to store in the DTED file header. The default is '**U**' for unclassified. Other valid values are '**C**' (confidential), '**S**' (secret), and '**R**' Restricted.
- **SPLIT_INTO_FOLDERS** - write each column out to separate folders by longitude. Use SPLIT_INTO_FOLDERS=**YES** to enable this behavior.

Lidar LAS/LAZ Fields

- **SAVE_HEIGHTS_ABOVE_GROUND** - specifies whether the exported LAS/LAZ file should contain raw elevations or a calculated 'height above ground'. Use SAVE_HEIGHTS_ABOVE_GROUND=**YES** to enable saving heights above ground. Note this requires the separate Lidar module to be licensed.
- LAS_VERSION - specifies what version of LAS file to write out. This would be **1.1**, **1.2**, **1.3**, or **1.4**. If you don't specify a version, the lowest version that will support all of the provided options will be used (typically 1.1 or 1.2).
- **VERT_CS_CODE** - specifies the vertical coordinate system (i.e. vertical datum) to store in the LAS file to specify what the elevations are referenced to. Use the *EPSG code*, like **5103** for NAVD88. If you don't specify a value and the source files used all use the same known system, that will be used. Note that no vertical datum conversion is done, this is just to supply metadata.
- **VERT_CITATION** - specifies the text description to store in the Lidar LAS file for the vertical coordinate system for the elevations. If nothing is supplied the default one (if any) for the supplied VERT_CS_CODE will be used.

- **FILE_SOURCE_ID** - specifies a File Source ID numeric value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.
- **GLOBAL_ENCODING** - specifies a Global Encoding numeric value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.
- **SYSTEM_ID** - specifies a System ID value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.
- **GEN_SOFTWARE** - specifies a Generating Software value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used. Otherwise 'Global Mapper' will be used.
- **FLIGHT_DATE** - specifies the flight date to store in the exported LAS file header. This can be either the *day of the current year* (value **1** to **366**) or *a common date format, including month, day, and year*. If not specified the current date will be used.
- **GUID** - specifies the GUID to store in the LAS header. The format should be like `GUID="{21EC2020-3AEA-4069-A2DD-08002B30309D}"`.
- **USE_8BIT_CLASSES** - specifies that full 8-bit values should be used to store the point classification, allowing values up to 255, even in LAS 1.1/1.2 files that only support classes up to 31 per the standard. Use with care as the files created using USE_8BIT_CLASSES=YES will not comply with the LAS standards if any classes over 31 are present.
- **SAVE_HEIGHTS_ABOVE_GROUND** - specifies whether the exported LAS/LAZ file should contain raw elevations or a calculated 'height above ground'. Use SAVE_HEIGHTS_ABOVE_GROUND=**YES** to enable saving heights above ground. Note this requires the separate Lidar module to be licensed.
- LAS_VERSION - specifies what version of LAS file to write out. This would be **1.1**, **1.2**, **1.3**, or **1.4**. If you don't specify a version, the lowest version that will support all of the provided options will be used (typically 1.1 or 1.2).
- **VERT_CS_CODE** - specifies the vertical coordinate system (i.e. vertical datum) to store in the LAS file to specify what the elevations are referenced to. Use the *EPSG code*, like **5103** for NAVD88. If you don't specify a value and the source files used all use the same known system, that will be used. Note that no vertical datum conversion is done, this is just to supply metadata.
- **VERT_CITATION** - specifies the text description to store in the Lidar LAS file for the vertical coordinate system for the elevations. If nothing is supplied the default one (if any) for the supplied VERT_CS_CODE will be used.
- **FILE_SOURCE_ID** - specifies a File Source ID numeric value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.
- **GLOBAL_ENCODING** - specifies a Global Encoding numeric value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.
- **SYSTEM_ID** - specifies a System ID value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.
- **GEN_SOFTWARE** - specifies a Generating Software value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used. Otherwise 'Global Mapper' will be used.
- **FLIGHT_DATE** - specifies the flight date to store in the exported LAS file header. This can be either the *day of the current year* (value **1** to **366**) or *a common date format, including month, day, and year*. If not specified the current date will be used.
- **GUID** - specifies the GUID to store in the LAS header. The format should be like `GUID="{21EC2020-3AEA-4069-A2DD-08002B30309D}"`.
- **USE_8BIT_CLASSES** - specifies that full 8-bit values should be used to store the point classification, allowing values up to 255, even in LAS 1.1/1.2 files that only support classes up to 31 per the standard. Use with care as the files created using USE_8BIT_CLASSES=YES will not comply with the LAS standards if any classes over 31 are present.

Arc ASCII Grid Fields

- **VOID_ELEV** - specifies the string to use for no data values. The **default** is **-9999.0**.
- **PRECISION** - specifies the number of digits of precision to include after the decimal. The default is an automatic determination based on the actual value. If you want to instead use just one digit after the decimal always, use PRECISION=**1**.

- **USE_CENTER_COORDS** - specifies that center coordinates rather than corner coordinates should be written to the Arc ASCII Grid header. Use USE_CENTER_COORDS=**YES** to enable.

GWS Windsim Fields

- **ELEV_LAYER** - filename or description of elevation layer(s) to export for Windsim GWS export. By default all compatible and exportable layers are exported. You can include multiple ELEV_LAYER parameters if you have multiple masks to search. Wildcards (* and ?) are supported. Hidden layers are not considered.

- **ROUGHNESS_LAYER** - filename or description of roughness grid layer(s) to export for Windsim GWS export. By default all compatible and exportable layers are exported. You can include multiple ROUGHNESS_LAYER parameters if you have multiple masks to search. Wildcards (* and ?) are supported. Hidden layers are not considered.

Other Format Specific Fields

- **QUAD_NAME** (USGS_DEM only) - specifies the quad name to place in the header of the USGS DEM file.

- **VERT_EXAG** (VRML only) - specifies the vertical exaggeration to use when creating the VRML file. Larger values result in a rougher terrain being generated while smaller values result in a smoother terrain. A value of **1.0** results in no vertical exaggeration at all (i.e. a nearly true to life representation). If you don't specify a value the currently selected vertical exaggeration value on the Vertical Options tab of the Configuation dialog will be used.

- **ALLOW_LOSSY** (GLOBAL_MAPPER_GRID only) - specifies whether or not a slight loss in precision is allowable in order to achieve better compression ratios. The default is **YES**, so turn only use lossless compression you need to specify a value of ALLOW_LOSSY=**NO**.

- **FORMAT** (SURFERGRID only) - determines if Surfer Grid export format is **ASCII**, **BINARY_V6**, or **BINARY_V7**. The default is ASCII if no format is specified.

- **EXPORT_ECEF** (DXF_3DFACE only) - specifies that the export should use ECEF (earth-centered earth-fixed, or geocentric) coordinates.

- **CREATE_BINARY** (STL only) - specifies that the STL file that is created will be a binary STL file rather than a (much larger) ASCII text STL file.

- **Z_UP** (STL only) - specifies that the STL file that is created will use a "Z-Up" orientation; that is, Y values in exported coordinates will represent elevations and Z values will represent latitudes.

- **NITF_USE_DECIMAL** (NITF only) - specifies that is a NITF file is exported with lat/lon coordinates that the ICORDS is set to D for decimal coordinates rather than G for DMS.

- **SRTM_LEVEL** (SRTM only)- specifies what SRTM level to use for SRTM HGT export. The values are **1** (1-arc-second resolution) and **3** (3-arc-second resolution). If not provided, a **default** of level 3 is used.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

- **Tiling / Gridding Parameters**

  See also Tiling/Gridding Export into Smaller Chunks

SAMPLES

Here is an example script showing 2 GWS exports:

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
// Do export with elev/roughness layers separately specified
EXPORT_ELEVATION TYPE="GWS" FILENAME="output.gws" \
ELEV_LAYER="*blue_springs*.dem*" ROUGHNESS_LAYER="Roughness *"
// Do export with automatic layer selection
EXPORT_ELEVATION TYPE="GWS" FILENAME="output_auto.gws"
```

EXPORT_METADATA

The EXPORT_METADATA command exports the metadata for a specified load layer. The following parameters are supported by the command.

- **FILENAME** - full path of file (must already be loaded) that you want to save the metadata for.
- **METADATA_FILENAME** - full path of new text file to create on disk containing the metadata for the specified layer.
- **ADD_LIDAR_STATS** - specifies whether or not statistics about a Lidar point cloud should be dumped to the file for Lidar point cloud layers. This includes all of the statistics that you see on the Statistics tab of the Metadata dialog for a Lidar layer. Use ADD_LIDAR_STATS=**YES** to enable.
- **APPEND_TO_FILE** - specifies that the metadata text should be appended to the file specified if it already exists rather than a new file being created. Use APPEND_TO_FILE=**YES** to enable.
- **HEADER** - specifies a header line to include before the metadata output by this statement. If you want to insert extra line breaks in the header, use the **\n** escape character.
- **ADD_BLANK_LINE** - specifies that a blank line will be added to the file if APPEND_TO_FILE=YES is added to the command and the file was not empty to start with. Use ADD_BLANK_LINE=**YES** to enable adding the blank line.

EXPORT_PACKAGE

The EXPORT_PACKAGE command exports all currently loaded raster, vector, and elevation data to a Global Mapper Package (GMP) file. The following parameters are supported by the command.

- **FILENAME** - full path to file to save the data to
- **EXPORT_LAYER** - filename or description of layer(s) to export. By default all compatible and exportable layers are exported. You can include multiple EXPORT_LAYER parameters if you have multiple masks to search. Wildcards (***** and *?*) are supported. Hidden layers are not considered.
- **SIMPLIFICATION** - specifies the simplification threshold to use when exporting the vector features. This specifies how far off a straight line (in the units of the current projection) that a point has to be before it is kept. Generally you should not specify a simplification value as the default just exports all vertices as they are. This is an option for advanced users only.
- **SINGLE_PRECISION** - specifies that vector coordinates should be written out as 4-byte single precision floating point numbers rather than the default 8-byte double precision floating point numbers. Use SINGLE_PRECISION=**YES** to enable single precision export, which will result in smaller files.
- **DISCARD_ATTRIBUTES** - specifies that the list of attribute-value pairs for each vector feature should not be written out. Use DISCARD_ATTRIBUTES=**YES** to enable this behavior.
- **VECTOR_ONLY** - specifies that only vector layers should be exported to the package file. Use VECTOR_ONLY=**YES** to enable.
- **SHAPE_TYPE** specifies the vector object type(s) (area, line, or point) to export. You can *specify a comma-delimited list of the following* (like SHAPE_TYPE="AREAS,LINES") or if you don't provide a value at all the default of all types will be exported:
  - **AREAS** - export area features
  - **LINES** - export line features
  - **POINTS** - export point features
- **KEEP_ALL_STYLES** - specifies that the full style of each feature should be written to the package file, even if it uses the current default for the type. Use KEEP_ALL_STYLES=**YES** to enable this. This is useful if you want features to look exactly the same on all systems and not be affected by changes to the default styling for types.
- **KEEP_NATIVE_PROJECTIO**N - specifies that each layer should be exported in the native projection of the layer rather than reprojected to the current projection. Use KEEP_NATIVE_PROJECTION=**YES** to enable this behavior.
- **COMBINE_VECTOR_LAYERS** - specifies that all vector data should be combined into a single layer within the package file. Use COMBINE_VECTOR_LAYERS=**YES** to enable this behavior.
- **OVERWRITE_EXISTING** - specifies that existing files should be overwritten. The default is OVERWRITE_EXISTING=**YES**, so use OVERWRITE_EXISTING=**NO** to skip exporting files that already exist.

- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

- **Tiling/Gridding Export into Smaller Chunks**

  See also "Gridding/Tiling Operations into Smaller Chunks" on page 124

EXPORT_PDF

The EXPORT_PDF command exports all currently loaded data to a PDF file. The following parameters are supported by the command.

- **FILENAME** - full path to file to save the data to
- **EXPORT_LAYER** - filename or description of layer(s) to export. By default all compatible and exportable layers are exported. You can include multiple EXPORT_LAYER parameters if you have multiple masks to search. Wildcards (* and ?) are supported. Hidden layers are not considered.
- **DPI** - specifies the DPI that the file is generated at.
- **EXPORT_SCALE** - specifies the scale to do the export at. For example to export at 1:50,000 scale, use `EXPORT_SCALE=50000`.
- **PDF_PAGE_SIZE** - specifies the name of the paper size to use. This should match one of the entries on the PDF export dialog, like **Letter**, **A0**, etc. If you would like to define a custom page size, use a format like `PDF_PAGE_SIZE="CUSTOM 12in x 18in"` (creates a 12 inch wide by 18 inch tall page or `PDF_PAGE_SIZE-E="CUSTOM 30cm X 60cm"` which creates a 30 cm by 80 cm page.
- **PDF_PAGE_ORIENTATION** - specifies the page orientation to use for the generated PDF file. The following values are supported:
  - **AUTO** - Automatically determine whether to use portrait or landscape mode based on export bounds aspect ratio.
  - **PORTRAIT**
  - **LANDSCAPE**
- **PDF_MARGINS** - specifies the margins to use in inches. The value should be a *comma-delimited list of numbers for the left, top, right, and bottom margins*. For example, `PDF_MARGINS="0.5,1.0,0.5,1.25"` would result in a 0.5 inch margin for the left and right, 1.0 inch for the top, and 1.25 inches for the bottom.
- **PDF_HEADER** - specifies the header to use
- **PDF_FOOTER** - specifies the footer to use
- **PDF_COMBINE_RASTERS** - specifies whether multiple raster layers should be combined into a single layer or kept separate. Use PDF_COMBINE_RASTERS=**YES** to combine them or PDF_COMBINE_RASTER_LAYERS=**NO** to keep separate.
- **PDF_FILL_PAGE** - specifies whether the PDF export should fill the page or be cropped to the exact bounds specified. Use PDF_FILL_PAGE=**YES** to enable or PDF_FILL_PAGE=**NO** to disable.
- **PDF_FONT_SCALE** - specifies the scale factor to apply to text. For example use `PDF_FONT_SCALE=2.0` to double the size of text.
- **PDF_SYMBOL_SCALE** - specifies the scale factor to apply to point symbols and icons. For example use `PDF_SYMBOL_SCALE=2.0` to double the size of symbols.
- **LAYER_ATTR** - specifies the attribute value to use from each feature for the layer name in the PDF file. The default is to use the feature description. See special Attribute Name parameter details for recognized values.
- **VECTOR_ONLY** - specifies that only vector layers should be exported to the PDF file. Use VECTOR_ONLY=**YES** to enable.
- **SAVE_GRID_LINES** - specifies that if grid line display is enabled that the grid lines should be saved. Specify SAVE_GRID_LINES=**NO** to disable this option. If it's not specified the grid lines will be saved if enabled.

- **SAVE_SCALE_AND_LEGEND** - specifies that the distance scale and elevation legend, if applicable and enabled for display on the Configuration dialog, should be exported to the generated PDF file. Specify SAVE_SCALE_AND_LEGEND=**NO** to disable this option. If it's not specified the current view settings will be used.
- **PDF_USE_ADOBE_EXT** - specifies that Adobe ISO 32000 Extensions should be used for georeference. Specify PDF_USE_ADOBE_EXT=**YES** to enable this option (default) or PDF_USE_ADOBE_EXT=**NO** to disable this option and use standard PDF georeferencing.
- **OVERWRITE_EXISTING** - specifies that existing files should be overwritten. The default is OVERWRITE_EXISTING=**YES**, so use OVERWRITE_EXISTING=**NO** to skip exporting files that already exist.
- **Specify Bounding Box for Operation**

  See also "Specify Bounds for Operation" on page 125
- **Tiling/Gridding Export into Smaller Chunks**

  See also "Gridding/Tiling Operations into Smaller Chunks" on page 124

EXPORT_PDF3D

The EXPORT_PDF3D command exports all currently loaded data to a 3D PDF file. The following parameters are supported by the command.

- **FILENAME** - full path to file to save the data to
- **EXPORT_LAYER** - filename or description of layer(s) to export. By default all compatible and exportable layers are exported. You can include multiple EXPORT_LAYER parameters if you have multiple masks to search. Wildcards (* and ?) are supported. Hidden layers are not considered.
- **PDF_PAGE_SIZE** - specifies the name of the paper size to use. This should match one of the entries on the PDF export dialog, like **Letter**, **A0**, etc. If you would like to define a custom page size, use a format like `PDF_PAGE_SIZE="CUSTOM 12in x 18in"` (creates a 12 inch wide by 18 inch tall page or `PDF_PAGE_SIZE="CUSTOM 30cm X 60cm"` which creates a 30 cm by 80 cm page.
- **PDF_PAGE_ORIENTATION** - specifies the page orientation to use for the generated PDF file. The following values are supported:
    - **AUTO** - Automatically determine whether to use portrait or landscape mode based on export bounds aspect ratio.
    - **PORTRAIT**
    - **LANDSCAPE**
- **PDF_MARGINS** - specifies the margins to use in inches. The value should be a comma-delimited list of numbers for the left, top, right, and bottom margins. For example, `PDF_MARGINS="0.5,1.0,0.5,1.25"` would result in a 0.5 inch margin for the left and right, 1.0 inch for the top, and 1.25 inches for the bottom.
- **PDF_HEADER** - specifies the header to use
- **PDF_FOOTER** - specifies the footer to use
- **PDF_COMBINE_RASTERS** - specifies whether multiple raster layers should be combined into a single node or kept separate. Use PDF_COMBINE_RASTERS=**YES** to combine them (default) or PDF_COMBINE_RASTERS=**NO** to keep separate.
- **PDF_COMBINE_ELEV_GRIDS** - specifies wheter multiple elevation grid layers should be combined into a single node or kept separate. Use PDF_COMBINE_ELEV_GRIDS=**YES** to combine them (default) or PDF_COMBINE_ELEV_GRIDS=**NO** to keep separate.
- **PDF_CREATE_IMAGE_DRAPED** - specifies whether to combine raster and elevation grid layers into a single image draped surface node. Use PDF_CREATE_IMAGE_DRAPED=**YES** to combine them (default) or PDF_CREATE_IMAGE_DRAPED=**NO** to keep separate.
- **PDF_LIDAR_AS_LINES** - specifies wether to render Lidar points as short line segments in PDF. This will improve appearance in some PDF readers. Specify PDF_LIDAR_AS_LINES=**YES** to enable this option (default) or PDF_LIDAR_AS_LINES=**NO** to disable this option and render as points.
- **PDF_USE_FILENAME_FOR_NODES** - specifies whether to use the source filename for exported node names. Specify PDF_USE_FILENAME_FOR_NODES=**YES** to enable this option or PDF_USE_FILENAME_FOR_NODES=**NO** to disable this option and assign default node names based on data type (raster, lidar, point, line, area).

- **PDF_USE_LDP** - specifies whether to use a Low Distortion Projection for exported data. This can impove accuracy when the range of coordinates is small compared to their values. Specify LDP=**YES** to enable this option (**default**) or PDF_USE_LDP=**NO** to disable this option.

- **PDF_USE_ADOBE_EXT** - specifies that Adobe ISO 32000 Extensions should be used for georeference. Specify PDF_USE_ADOBE_EXT=**YES** to enable this option (default) or PDF_USE_ADOBE_EXT=**NO** to disable this option and use standard PDF georeferencing.

- **OVERWRITE_EXISTING** - specifies that existing files should be overwritten. The default is OVERWRITE_EXISTING=**YES**, so use OVERWRITE_EXISTING=**NO** to skip exporting files that already exist.

- **Specify Bounding Box for Operation**

- **Tiling/Gridding Export into Smaller Chunks**

EXPORT_RASTER

The EXPORT_RASTER command exports all currently loaded raster, vector, and elevation data to a file. The following parameters are supported by the command.

- **FILENAME** - full path to file to save the data to. When using additional name parameters with Tiling/Gridding or Polygon Cropping, this can specify just a directory, or the prefix and/or extension for the filename.

- **EXPORT_LAYER** - filename or description of layer(s) to export. By default all compatible and exportable layers are exported. You can include multiple EXPORT_LAYER parameters if you have multiple masks to search. Wildcards (* and *?*) are supported. Hidden layers are not considered.

- **TYPE**

  type of raster file we're exporting to

    - **FIRST_LOADED** - export to the same type as the first loaded file
    - **LAST_LOADED** - export to the same type as the last loaded file
    - **ADRG** - export to an ADRG data set. See supported ADRG fields below.
    - **ASRP** - export to an ASRP data set. See supported ASRP fields below.
    - **BIL** - export to a band interleave (BIL) format file.
    - **BIP** - export to a band interleaved pixel (BIP) format file.
    - **BMP** - export to a Windows bitmap (BMP) format file.
    - **BSB** - export to a BSB/KAP chart file.
    - **BSQ** - export to a band sequential (BSQ) format file.
    - **ECW** - export to an ECW format file.
    - **ERDAS** - export to an Erdas Imagine IMG format file.
    - **GEOTIFF** - export to a GeoTIFF format file.
    - **IDRISI_RASTER** - export to an Idrisi raster format file.
    - **JPEG** - export to a JPG format file.
    - **JPEG2000** - export to a JPEG2000 format file.
    - **KML** - export to a raster KML/KMZ format file for display in Google Earth.
    - **NITF** - NITF format imagery
    - **PCX** - export to a PCX format file.
    - **PNG** - export to a PNG format file.
    - **RPF** - export to a RPF (CADRG or CIB) data set. See required RPF fields below.
    - **XY_COLOR** - export to a XY color format file.

- **SPATIAL_RES** - specifies spatial resolution. Defaults to the minimum spatial resolution of the two layers if not specified. Should be formatted as *x_resolution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to export at 30 meter spacing, the parameter/value pair would look like `SPATIAL_RES=30.0,30.0`. You can also specify as a *percentage of the default resolution* by adding a percent. For example to get half the detail your double the spatial resolution value, so you would use `SPATIAL_RES="200%,200%"`.

- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will automatically be converted to the current view/ export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.

- **PIXEL_SIZE** - specifies the desired size in pixels of your export. Use this instead of SPATIAL_RES if you know exactly how many pixels in size your export should be. The format is PIXEL_SIZE="*widthxheight*". For example, to make your export have dimensions of 1024 pixels wide by 768 pixels tall, use `PIXEL_SIZE-E="1024x768"`.

- **PIXEL_SIZE_MAX** - specifies the maximum desired size in pixels of your export. The format is the same as the PIXEL_SIZE parameter, but specifies the maximum dimension on either side. If the dimensions result in non-square pixels, a smaller pixel count will be used in width or height so that the pixel aspect ratio is maintained.

- **FORCE_SQUARE_PIXELS** - if this value is set to **YES**, the spatial resolution of the resultant raster file will be set so that the x and y pixel size are the same, with the minimum default size being used for both.

- **SAMPLING_METHOD** - specifies the sampling method to use when resampling this layer. The following values are supported:
  - **DEFAULT** - Use either automatic resampling based on export or layer resampling, depending on setting of global flag about whether to resample on export
  - **AUTO** - Automatically select a resampling method based on how the export resolution and bounds compare to the original layout for a layer. For example if you export to a lower resolution a box averager of appropriate size may be used automatically
  - **LAYER** - Use the sampling method set for each layer
  - The list of SAMPLING_METHOD values for the "IMPORT" on page 35 command can also be specified to use a particular sampling method for all layers being exported

    Shared IMPORT SAMPLING_METHOD values

    See "SAMPLING_METHOD (elevation and raster only) - specifies the sampling method to use when resampling this layer. " on page 39

- **PALETTE**

  Specifies the palette/image type to use. If not specified, a 24-bit RGB image will be generated.

  - **KEEP_SOURCE** - The exported file will use the some color configuration (if possible) as the file being exported. Note that this option is only available if you have only a single layer loaded for export.
  - **OPTIMIZED** (BMP, ERDAS, GEOTIFF, and PNG only) - The palette generated will be an optimal mix of up to 256 colors that will closely represent the full blend of colors in the source images. This option will generate the best results, but can more than double the export time required if any high color images are present in the export set.
  - **HALFTONE** (BMP, ERDAS, GEOTIFF, and PNG only) - use a 256-color halftone palette spread over the color spectrum
  - **DOQ_DRG** (BMP, ERDAS, GEOTIFF, and PNG only) - use a 256-color palette optimized for combined grayscale DOQs and USGS DRGs
  - **DRG** (BMP, ERDAS, GEOTIFF, and PNG only) - use a 256-color palette optimized for the colors found in USGS DRGs
  - **GRAYSCALE** - use a 256-color grayscale palette
  - **BW** (BMP, GEOTIFF only) - creates a black and white, 1-bit per pixel image
  - **BLACKISMIN** (GEOTIFF only) - creates an 8-bit per pixel grayscale image with no color map stored in the image. Black will be stored as zero with varying shades of gray up to white with a value of 255.
  - **WHITEISMIN** (GEOTIFF only) - creates an 8-bit per pixel grayscale image with no color map stored in the image. White will be stored as zero with varying shades of gray up to black with a value of 255.
  - **JPG** (GEOTIFF only) - creates a 24-bit RGB JPG-in-TIFF image. Note that while this creates a highly compressed GeoTIFF file, many software packages do not recognize JPG-in-TIFF format files.
  - **MULTIBAND** (BIL, BIP. BSQ, GEOTIFF, JPEG2000, and NITF only) - creates a multi-band image file with any number of bands of data. This is useful for multi-spectral imagery. Either 8- or 16-bits per band will be used depending on what is available in the input data. Use the NUM_BANDS parameter to specify how many bands to use. You can override the default band setting using the BAND_EXPORT_SETUP parameter (described below).
  - *Custom palette filename* - you can also pass in the full path to a .pal file containing a custom palette to use for the export.

- **NUM_BANDS** - specifies how many bands of data to export for a PALETTE=MULTIBAND export. If you don't specify a value for this the band count will be the maximum available for any of the loaded layers.

- **BAND_BIT_DEPTH** - specifies how many bits to use for each band of data for a PALETTE=MULTIBAND export. If you don't specify a value for this the highest bit depth of any of the input data layers will be used. The valid values are BAND_BIT_DEPTH=**8**, BAND_BIT_DEPTH=**16**, or BAND_BIT_DEPTH=**32**.

- **BAND_EXPORT_SETUP** - allows you to override the default band assignment for a MULTIBAND export. Use the following format to specify what band from what layer to use for a given export band: *<output_band>*?*<input_band>*?*<layer_filename>* . So for example to assign the 4th (infrared) band in an export from the 1st (red) band in a previously loaded file name C:\data\input_file.tif, use the following parameter: `BAND_EXPORT_SETUP="4?1?c:\data\input_file.tif"`. Note that you would include a separate BAND_

EXPORT_SETUP parameter for each output band that you want to setup. If you leave off the filename then all loaded data will be considered as input, with just the input-to-output band assignment being updated.

- **INC_VECTOR_DATA** - specifies whether or not loaded vector data should be rendered and exported to the generated image. Use a value of **YES** to indicate that vector data should be used. Any other value will result in vector data NOT being saved to the file.

- **FILL_GAPS** - specifies that small gaps in between and within the data sets being exported will be filled in by interpolating the surrounding data to come up with a color for the point in question. This option is off by default, specify FILL_GAPS=**YES** to turn it on.

- **ONLY_GENERATE_METADATA** (GEOTIFF, JPEG, and PNG only) - specifies that only metadata files like world files, TAB files, and PRJ files should be created for this file. This is useful for things like generating world and TAB files from GeoTIFF files without doing a whole new export. Just make the output filename the same as the loaded file to create the metadata for.

- **SAVE_SCALE_AND_LEGEND** - specifies that the distance scale and elevation legend, if applicable and enabled for display on the Configuration dialog, should be exported to the generated raster file. Specify SAVE_SCALE_AND_LEGEND=**YES** to enable this option.

- **BG_TRANSPARENT** (ECW, GEOTIFF, JPEG2000 and PNG only) - specifies that any areas of no data/-background should be marked as transparent. Use BG_TRANSPARENT=**YES** to enable.

- **OVERWRITE_EXISTING** - specifies that existing files should be overwritten. The default is OVERWRITE_EXISTING=**YES**, so use OVERWRITE_EXISTING=**NO** to skip exporting files that already exist.

- **EXPORT_SCALE** - specifies the scale to do the export at. You must also specify a DPI value in order to use the EXPORT_SCALE parameter. For example to export at 1:50,000 scale, use EXPORT_SCALE=50000.

- **DPI** (GEOTIFF, BMP, and JPG only or with EXPORT_SCALE parameter) - specifies the DPI (dots per inch) value to save in the generated file(s). For example, use DPI=300 to specify that the DPI for this file is 300. By default no DPI value will be written out.

Projection Files

- **GEN_WORLD_FILE** - specifies that a world file should be generated in addition to the image file. Use GEN_WORLD_FILE=**YES** to turn on.

- **GEN_TAB_FILE** (GEOTIFF and PNG only) - specifies that a MapInfo TAB file should be generated in addition to the image file. Use GEN_TAB_FILE=**YES** to turn on.

- **GEN_PRJ_FILE** - specifies that a projection (PRJ) file should be generated in addition to the data file. Use GEN_PRJ_FILE=**YES** to turn on.

- **GEN_AUX_XML_FILE** - specifies that an ESRI .aux.xml projection file should be generated in addition to the data file. Use GEN_AUX_XML_FILE=**YES** to turn on.

- **GEN_OZI_MAP_FILE** - specifies that an OziExplorer .map file should be generated in addition to the data file. Use GEN_OZI_MAP_FILE=**YES** to turn on.

- **GEN_ERS_FILE** - specifies that an ERMapper .ers header file should be generated in addition to the data file. Use GEN_ERS_FILE=**YES** to turn on.

GeoTIFF Fields

- **COMPRESSION** - specifies the type of compression to use for the generated TIFF file. If you do not provide a compression value the default compression for each type will be used. The following values are supported:
    - **NONE** - Do not compress the output.
    - **LZW** - Use LZW (lossless) compression on the output.
    - **JPEG** - Use JPEG-in-TIFF (lossy) compression. Only works for 24-bit RGB output. Use QUALITY parameter to set quality setting.
    - **PACKBITS** - Use Packbits (lossless) compression. Only works for 8-bit palette-based output.
    - **DEFLATE** - Use Deflate/ZIP (lossless) compression on the output.

- *USE_LZW* [DEPRECATED - use COMPRESSION instead]- specifies that LZW compression should be used for this RGB or palette-based GeoTIFF file. LZW compression typically results in much smaller files than the default compression, but there may be some software packages that do not yet support LZW-encoded GeoTIFF files. Specify USE_LZW=YES to enable LZW compression.

- **QUALITY** (JPEG or GEOTIFF only) - specifies the quality setting to use when generating the image. Valid values range from **1** to **100**, with 1 generating the lowest quality image and 100 generating the highest quality image. If no QUALITY setting is present, a **default** value of **75** is used which generates a very high

quality image that is still highly compressed. Note that if a different quality value has been selected on the GeoTIFF export dialog in the user interface the last used value there will be the default.

- **TILE_SIZE** - specifies that the GeoTIFF file should be exported with a tiled organization and use the specified tile size. This tile size should be between **8** and **2048**. Typical values are **64**, **128**, and **256**.
- **DISABLE_BIGTIFF** - use to disable the automatic creation of BigTIFF-format files for very large exports. Use DISABLE_BIGTIFF=**YES** to disable the automatic BigTIFF support.
- **TIFF_COPYRIGHT** - specify text to store in TIFFTAG_COPYRIGHT tag.
- **TIFF_DATETIME** - specify text to store in TIFFTAG_DATETIME tag.
- **TIFF_DOC_NAME** - specify text to store in TIFFTAG_DOCUMENTNAME tag.
- **TIFF_GT_CITATION** - specify text to store in GeoTIFF GTCitationGeoKey GeoTIFF tag.
- **TIFF_IMAGE_DESC** - specify text to store in TIFFTAG_IMAGEDESCRIPTION tag.
- **TIFF_PCS_CITATION** - specify text to store in GeoTIFF PCSCitationGeoKey GeoTIFF tag.
- **TIFF_NO_GTIFF_HEADER** - don't embed a GeoTIFF header in the file. Use TIFF_NO_GTIFF_ HEADER=**YES** to disable write of header.

GeoTIFF Fields

- **COMPRESSION** - specifies the type of compression to use for the generated TIFF file. If you do not provide a compression value the default compression for each type will be used. The following values are supported:
  - **NONE** - Do not compress the output.
  - **LZW** - Use LZW (lossless) compression on the output.
  - **JPEG** - Use JPEG-in-TIFF (lossy) compression. Only works for 24-bit RGB output. Use QUALITY parameter to set quality setting.
  - **PACKBITS** - Use Packbits (lossless) compression. Only works for 8-bit palette-based output.
  - **DEFLATE** - Use Deflate/ZIP (lossless) compression on the output.
- *USE_LZW* [DEPRECATED - use COMPRESSION instead]- specifies that LZW compression should be used for this RGB or palette-based GeoTIFF file. LZW compression typically results in much smaller files than the default compression, but there may be some software packages that do not yet support LZW-encoded GeoTIFF files. Specify USE_LZW=YES to enable LZW compression.
- **QUALITY** (JPEG or GEOTIFF only) - specifies the quality setting to use when generating the image. Valid values range from **1** to **100**, with 1 generating the lowest quality image and 100 generating the highest quality image. If no QUALITY setting is present, a **default** value of **75** is used which generates a very high quality image that is still highly compressed. Note that if a different quality value has been selected on the GeoTIFF export dialog in the user interface the last used value there will be the default.
- **TILE_SIZE** - specifies that the GeoTIFF file should be exported with a tiled organization and use the specified tile size. This tile size should be between **8** and **2048**. Typical values are **64**, **128**, and **256**.
- **DISABLE_BIGTIFF** - use to disable the automatic creation of BigTIFF-format files for very large exports. Use DISABLE_BIGTIFF=**YES** to disable the automatic BigTIFF support.
- **TIFF_COPYRIGHT** - specify text to store in TIFFTAG_COPYRIGHT tag.
- **TIFF_DATETIME** - specify text to store in TIFFTAG_DATETIME tag.
- **TIFF_DOC_NAME** - specify text to store in TIFFTAG_DOCUMENTNAME tag.
- **TIFF_GT_CITATION** - specify text to store in GeoTIFF GTCitationGeoKey GeoTIFF tag.
- **TIFF_IMAGE_DESC** - specify text to store in TIFFTAG_IMAGEDESCRIPTION tag.
- **TIFF_PCS_CITATION** - specify text to store in GeoTIFF PCSCitationGeoKey GeoTIFF tag.
- **TIFF_NO_GTIFF_HEADER** - don't embed a GeoTIFF header in the file. Use TIFF_NO_GTIFF_ HEADER=**YES** to disable write of header.

KML/KMZ Fields

- **KML_MIN_LOD_PIXELS** - specifies how large layer has to be in pixels before it will show up in Google Earth.
- **KML_MAX_LOD_PIXELS** - specifies how large layer has to be in pixels before it will stop showing up in Google Earth. The default value of -1 which means that a layer will never go away once it is displayed.
- **KML_FADE_EXTENT_MIN** - specifies at what number of pixels in size that the image will start fading out. This value should be between KML_MIN_LOD_PIXELS and KML_MAX_LOD_PIXELS. The fade will be such

that the image is 100% opaque at KML_FADE_EXTENT_MIN and completely transparent at KML_MIN_LOD_ PIXELS.

- **KML_FADE_EXTENT_MAX** - specifies at what number of pixels in size that the image will start fading out. This value should be between KML_MIN_LOD_PIXELS and KML_MAX_LOD_PIXELS. The fade will be such that the image is 100% opaque at KML_FADE_EXTENT_MAX and completely transparent at KML_ MAX_LOD_PIXELS.

- **KML_RASTER_FORMAT** - specified which raster image format to use when creating tiles for KML/KMZ files. The valid options are **JPG**, **PNG**, and **TIFF**. For example, add `KML_RASTER_FORMAT=JPG` to use JPG format files.

- **KML_SUPER_OVERLAY** - specifies that the data should be automatically gridded into "super overlays" to allow displaying large quantities of data in Google Earth. Use KML_SUPER_OVERLAY=**YES** to enable this behavior.

- **KML_TILE_SIZE** - if data is being automatically gridded into "super overlays", this specifies the size of tiles to use for gridding. The **default** tile size is **1024**. To change this for example to 512x512, use KML_ TILE_SIZE=**512**.

- **KML_ZOOM_SCALE_FACTOR** - if data is being automatically gridded into "super overlays", this specifies the multiplier to use when creating zoomed out pyramid layers. The **default** value of **2** makes each successive zoom level 1/2 the resolution of the previous one until everything fits in a single tile. To change this to making each layer 1/3rd the resolution of the previous one, use `KML_ZOOM_SCALE_FACTOR=3`.

- **KML_GEN_INDEX_FILE** - specifies that an _index.kml file should be generated in the target folder which is an index to the individual KML/KMZ tiles that were exported.

- **KML_RASTER_ALTITUDE_MODE**- specifies how the altitude is interpreted.

    Valid values are:
    - **clampToSeaFloor** (default) - The overlay will be draped over the sea floor. If the point is on land rather than at sea, the overlay will be positioned on the ground.
    - **clampToGround** - Indicates to drape the overlay over the terrain.

BSB Fields

- **CHART_NAME** - Name of chart

- **CHART_NUMBER** - Chart number

- **CHART_SCALE** - Denominator of scale, like `CHART_SCALE=5000` for a 1:5,000 scale chart.

- **CHART_EDITION_DATE** - Chart edition date, like `CHART_EDITION_DATE="04/21/2014"` for April 21, 2014.

- **CREATE_BSB_FILE** - Flag to control whether or not a BSB file is created alongside KAP. Use CREATE_ BSB_FILE=**NO** to disable.

- **INC_POLYNOMIAL** - Flag to control whether or not the polynomial coordinate transformation is included in the KAP file. Use INC_POLYNOMIAL=**NO** to disable

RPF (CADRG/CIB) Fields

- **FILENAME** - Full path and name of a.toc file at root of data set export.

- **MAP_NAME** - The map name, usually a 6-character name

- **SERIES** - The chart series 2-letter code from [Section 5.1.4, MIL-STD-2411-1]. The following list includes some commonly supported values (note that any 2-letter code is supported):
    - **GN** - 1:5M Scale GNC (Global Navigation Chart)
    - **JN** - 1:2M Scale JNC (Jet Navigation Chart)
    - **TP** - 1:500K Scale TPC (Tactical Pilotage Chart)
    - **I1** - 10m Resolution CIB Imagery
    - **I2** - 5m Resolution CIB Imagery
    - **I3** - 2m Resolution CIB Imagery
    - **I4** - 1m Resolution CIB Imagery
    - **I5** - 0.5m Resolution CIB Imagery

- **SCALE** - The scale to export at. In most cases the SERIES implies the scale so this value is ignored, but it you use a SERIES with a variable scale this is required.

- **PRODUCER_CODE** - The *numeric producer code ID* from [MIL-STD-2411-1, Section 5.2.1), like **1** for AFACC (Air Force Air Combat Command) *or the abbreviation* (like **AFACC**).

- **AUTHOR** - The author to store in the NITF file header in the Originator's Name field ((MIL-STD-2500A 5.2)

- **SECURITY_CLASS** - The *1-character security classification* from [MIL-STD-2411-1, Section 5.1.8). The **default** is SECURITY_CLASS=**U** for unclassified.
- **SECURITY_COUNTRY** - The *2-character security country code* from [MIL-STD-2411-1, Section 5.1.7). The **default** is SECURITY_COUNTRY=**US** for the US.
- **SECURITY_MARKING** - The *2-character security marking* from [MIL-STD-2411-1, Section 5.1.9). The **default is** SECURITY_MARKING=**uu** for unclassified.
- **VERSION** - The file version. The **default** is **1** if you don't provide a value.
- **WRITE_EMPTY_FRAMES** - Specifies that all frames within the bounds should be written out, even if all pixels in the frame are transparent. Add WRITE_EMPTY_FRAMES=**YES** to enable this behavior.
- **MAX_COLORS** - specifies the maximum number of colors to use in the palette when compressing the CADRG/CIB. By **default** the maximum of **216** is used, but you can specify a smaller value to reduce the colors and perhaps maintain more sharpness in the compressed result.

ADRG/ASRP Fields

- **FILENAME** - Full path and name of TRANSH01.THF file at root of data set export.
- **MAP_NAME** - The map name, usually a 6-character name
- **VERSION** - The file version. The **default** is **12** for ASRP 1.2. The value of 11 is also supported for v1.1.

Other Format Specific Parameters

- **QUALITY** (JPEG or GEOTIFF only) - specifies the quality setting to use when generating the image. Valid values range from **1** to **100**, with 1 generating the lowest quality image and 100 generating the highest quality image. If no QUALITY setting is present, a **default** value of **75** is used which generates a very high quality image that is still highly compressed. Note that if a different quality value has been selected on the GeoTIFF export dialog in the user interface the last used value there will be the default.
- **TARGET_COMPRESSION** (ECW and JPEG2000 only) - specifies the target compression ratio to use when creating the ECW image. The default value is 10 which strikes a good balance between image quality and compression. The higher the value, the smaller the resulting image will be at the expense of image quality.
- **TILE_SIZE** (JPG2000 only) - specifies that the JPEG2000 file should internally use tile organization with the given tile size. By default an internal tile organization of size 1024x1024 is used. Use TILE_SIZE=**0** to disable tile organization.
- **ADD_OVERVIEW_LAYERS** (ERDAS only) - specifies that overview (pyramid) layers should be generated for the export. Use ADD_OVERVIEW_LAYERS=**YES** to enable.
- **BLOCK_SIZE** (ERDAS only) - specifies the block size to use for the export. The **default** is BLOCK_SIZE=**64**.
- **COMPRESS_OUTPUT** (ERDAS only) - specifies whether or not the exported file should be compress. The **default** is COMPRESS_OUTPUT=**YES**.
- **COORD_DELIM** (XY_COLOR only) - specifies the delimiter between coordinates
    - **COMMA** - coordinates are separated by commas
    - **SEMICOLON** - coordinates are separated by semicolons
    - **SPACE** - coordinates are separated by space characters
    - **TAB** - coordinates are separated by tab characters
- **USE_BAND4_AS_ALPHA** (ECW only) - specifies that the ECW file should store values from the 4th band of loaded layers to the alpha channel rather than using the alpha channel as on/off values. Use USE_BAND4_AS_ALPHA=**YES** to enable. This is an advanced option that allows some rudimentary multi-band ECW support.
- **SAVE_FULL_ALPHA** (PNG Only) - Specifies that the full range of alpha values from the top-most raster layer at each location should be stored for RGB PNG files with an alpha channel rather than just 0 (transparent) or 255 (opaque). Disabled by default, use SAVE_FULL_ALPHA=**YES** to enable.
- **NO_PROJ_HEADER** (ECW or JPEG2000 only) - specifies that no projection or datum information should be written to the file. Use NO_PROJ_HEADER=**YES** to indicate that the projection should not be written.
- **NITF_USE_DECIMAL** (NITF only) - specifies that is a NITF file is exported with lat/lon coordinates that the ICORDS is set to D for decimal coordinates rather than G for DMS.
- **Specify Bounding Box for Operation**

- **Cropping to Polygons/Areas**

  See also "Cropping Operations to Polygons/Areas" on page 123

- **Tiling / Gridding**

  See also "Gridding/Tiling Operations into Smaller Chunks" on page 124

EXPORT_VECTOR

The EXPORT_VECTOR command exports all currently loaded vector data to a file. The following parameters are supported by the command.

- **FILENAME** - full path to file to save the data to

- **EXPORT_LAYER** - filename or description of layer(s) to export. By default all compatible and exportable layers are exported. You can include multiple EXPORT_LAYER parameters if you have multiple masks to search. Wildcards (* and ?) are supported. Hidden layers are not considered.
  TYPE

  type of vector file we're exporting to

  - **FIRST_LOADED** - export to the same type as the first loaded file
  - **LAST_LOADED** - export to the same type as the last loaded file
  - **ANUDEM_CONTOUR** - export lines with elevation to an AnuDEM contour .gen format file.
  - **ARC_UNGENERATE** - export line and area features to an Arc Ungenerate format file.
  - **ARC_UNGENERATE_POINTS** - export point features to an Arc Ungenerate format file.
  - **CDF** - export to a Geographix CDF format file.
  - **COLLADA** - export to a Collada (DAE) 3D model format file.
  - **CSV** - export point features to a CSV format file.
  - **DELORME_DRAWING** - export features to a DeLorme drawing text file
  - **DELORME_TRACK** - export line features to a DeLorme track text file
  - **DELORME_WAYPOINT** - export point features to a DeLorme waypoint text file
  - **DGN** - export to a DGN v8 file.
  - **DLGO** - export to a native format USGS DLG-O file.
  - **DWG** - export to an AutoCAD DWG format file.
  - **DXF** - export to an AutoCAD DXF format file.
  - **GARMIN_TRK** - exports lines to a Garmin TRK (PCX5) format file.
  - **GARMIN_WPT** - exports names points to a Garmin WPT (PCX5) format file.
  - **GEOJSON** - exports area, line, and point features to a GeoJSON (JavaScript Object Notation) file.
  - **GPX** - exports line and point features to a GPX (GPS eXchange Format) file.
  - **INROADS** - exports to the InRoads ASCII format.
  - **KML** - export to a KML or KMZ format file.
  - **LANDMARK_GRAPHICS** - export to a Landmark Graphics format file.
  - **LIDAR_LAS** - export to a Lidar LAS/LAZ file. Use .laz in filename to get LasZip.
  - **LOWRANCE_USR** - export to a Lowrance USR format file.
  - **MAPGEN** - export to a MapGen format file.
  - **MAPINFO** - export to a MapInfo MIF/MID or TAB/MAP format file.
  - **MATLAB** - export to a MatLab format file.
  - **MOSS** - export line and area features to a MOSS format file.
  - **NIMA_ASC** - export to a NIMA ASC format file.
  - **OBJ** - export to a Wavefront OBJ 3D model format file.
  - **OSM** - export to a OpenStreetMap (OSM) XML format file.
  - **OV2** - export to a TomTom OV2 format file.
  - **PLATTE_RIVER** - export to a Platte River ASCII Digitizer format file.
  - **PLS_CADD** - export to a PLS-CADD format file.
  - **PLY** - export to a PLY 3D model format file.
  - **POLISH_MP** - export to a Polish MP format file.
  - **SEGP1** - export to a SEGP1 format file.
  - **SHAPEFILE** - export to an ESRI Shapefile format file.
  - **SIMPLE_ASCII** - export to a simple ASCII text/XYZ/distance/z file.
  - **SOSI** - exports area, line, and point features to a SOSI (Norwegian Data) file.
  - **STL** - export to an STL 3D model format file.
  - **SURFER_BLN** - export to a Surfer BLN format file.
  - **SVG** - export to a Scalable Vector Graphic (SVG) format file.

- **TSUNAMI_OVR** - export to a Tsunami OVR format file.
- **WASP_MAP** - export to a WAsP .map format file (line features only).
- **XYZI** - export to a XYZI (XYZ + intensity format file)
- **ZMAP_ISOMAP_LINE** - export to a ZMap+ IsoMap Line format file (line features only).
- **ZMAP_XYSEGID** - export to a ZMap+ XYSegId format file (area and line features only).

- **SAVE_GRID_LINES** - specifies that if grid line display is enabled that the grid lines should be saved. Specify SAVE_GRID_LINES=**NO** to disable this option. If it's not specified the the grid lines will be saved if enabled.

- **GEN_PRJ_FILE** - specifies that a projection (PRJ) file should be generated in addition to the output file. Set this to YES to cause the projection file to be generated. Leaving out this parameter or setting it to anything but **YES** will cause no PRJ file to be generated.

- **GEN_AUX_XML_FILE** - specifies that an ESRI .aux.xml projection file should be generated in addition to the data file. Use GEN_AUX_XML_FILE=**YES** to turn on.

- **OVERWRITE_EXISTING** - specifies that existing files should be overwritten. The default is OVERWRITE_EXISTING=**YES**, so use OVERWRITE_EXISTING=**NO** to skip exporting files that already exist.

- **SHAPE_TYPE** (works for any, required for SHAPEFILE only) - specifies the vector object type (area, line, or point) to export. For formats other than SHAPEFILE if you don't provide a value then all available features will be exported. For the SHAPEFILE format you must specify exactly one of the below. For other formats you can specify a comma-delimited list of the following (like SHAPE_TYPE="AREAS,LINES"):
    - **AREAS** - export area features
    - **LINES** - export line features
    - **POINTS** - export point features

- **Tiling/Gridding Export into Smaller Chunks**

    See also "Gridding/Tiling Operations into Smaller Chunks" on page 124

- **Specify Bounding Box for Operation**

    See also "Specify Bounds for Operation" on page 125

- **Cropping to Polygons/Areas**

    See also "Cropping Operations to Polygons/Areas" on page 123

Splitting Exports by Attribute Parameters

- **SPLIT_BY_ATTR** - specifies that the export should generate a separate file for each set of attributes values in the input data. Use the FILENAME_ATTR and/or FILENAME_ATTR_LIST and FILENAME_INCLUDE_ATTR_NAME parameters to control which attributes are compared to and in what order and how the filename is generated from those attributes and their values. Use SPLIT_BY_ATTR=**YES** to split your export so that all features with the same values for each of the specified attributes is in the same file.

- **FILENAME_ATTR** - contains a single attribute to use when naming files exported when using the SPLIT_BY_ATTR=YES parameter. If you would like to filter the results to only where an attribute has a specified value, do that with an equal sign, like *FILENAME_ATTR="<Feature Name>=My Label"*. You can also use **!=** rather than **=** to match on any feature with an attribute value not equal to the specified value. You can also embed the value of the attribute in the export path (or in any parameter of the command) by adding *%SPLIT_ATTR%* inside the command. For example use `FILENAME="c:\my_path\%SPLIT_ATTR%\prefix.ext"` to export each file inside a folder that includes the split attribute value.

- **FILENAME_ATTR_LIST** - contains a *comma-delimited list of attributes to use* when naming files exported when using the SPLIT_BY_ATTR=YES parameter. If you would like to filter the results to only where an attribute has a specified value, do that with an *equal sign*, like `FILENAME_ATTR_LIST="<Feature Name>=My Label,ATTR_1=My Attr Val"`. You can also use **!=** rather than **=** to match on any feature with an attribute value not equal to the specified value. If you need to match on a value that may contain a comma, use FILENAME_ATTR instead.

- **FILENAME_INCLUDE_ATTR_NAME** - specifies that the attribute name specified in the FILENAME_ATTR_LIST parameter should be included as part of the filename when using SPLIT_BY_ATTR=YES to split your export into a separate file for each set of attributes.

- **MAKE_FNAME_LOWER** - specifies that the generated filename should consist only of lower-case letters. Typically used along with SPLIT_BY_ATTR=YES when getting attribute values that are upper case and you want the filename to be lower case.

Shapefile Parameters

- **SPLIT_BY_LAYER** (SHAPEFILE only) - specifies that the export should generate a separate Shapefile for each layer/type in the input data
- **EXPORT_MEASURES**- specifies that 3D line and point objects should be export as PolyLineM and PointM features (respectively) rather than as PolyLineZ and PointZ features. Use EXPORT_MEASURES=**YES** to enable.
- **GEN_MULTI_PATCH** (SHAPEFILE only) - specifies that area features exported to a Shapefile should be stored as multi-patch features rather than as areas. Use GEN_MULTI_PATCH=**YES** to enable.
- **DISCARD_EMPTY_ATTRS** (SHAPEFILE only) - specifies whether or not to include attributes in the DBF file even if all encountered values are empty. By default these attributes are included, add DISCARD_ EMPTY_ATTRS=**YES** to discard them.
- **ALLOW_COMMA_DECIMAL** - specifies whether or not a comma character can act as a decimal sep- arator. Use ALLOW_COMMA_DECIMAL=**YES** to allow a comma as the decimal within string values. If not specified only periods will be allowed for decimals when determining the type for a field in the exported DBF file.
- **INC_MAP_NAME_ATTR** - specifies whether or not the name of the map file that a feature came from should be added as an attribute to the DBF files exported with the Shapefile. Use INC_MAP_NAME_ ATTR=**YES** to enable.
- **INC_STYLE_ATTRS** - specifies whether or not attributes for the feature styles of each feature should be added as attributes to the DBF files exported with the Shapefile This is disabled by default, use INC_ STYLE_ATTRS=**YES** to enable.
- **CODE_PAGE** - specifies the code page to use for the Shapefile export. The default is ANSI (**1252**). Use the *code page number*, or the text **UTF-8** (number 65001).
- **GEN_3D_FEATURES** - specifies that 3D line and point objects should be created in the exported file. Set this to **YES** to cause the the 3D features to be generated. Leaving out this parameter or setting it to any- thing but YES results in the normal 2D objects. The elevation stored for each vertex/point will be the first of the following that is available:
  - The elevation associated with the vertex/point in question.
  - The elevation associated with the entire area/line/point being exported. For example, the elevation of a contour line or spot elevation.
  - The first elevation obtained by searching the loaded elevation layers at the position of the ver- tex/point.

  A value of 0.0 will be used if no elevation could not be obtained via any of the prior methods.
- **INC_ELEV_ATTR** - specifies whether or not the elevation of a feature should be added as an attribute to the exported files. Use INC_ELEV_ATTR=**YES** to enable or INC_ELEV_ATTR=**NO** to disable. This is **enabled by default**.
- **INC_LAYER_ATTR** - specifies whether or not the layer (description) of a feature should be added as an attribute to the DBF files exported with the Shapefile. Use INC_LAYER_ATTR=**YES** to enable or INC_ LAYER_ATTR=**NO** to disable. This is **enabled by default**.
- **CREATE_EXTRUDE_FACES** - specifies whether or not to create 3D areas for the sides and bottom of extruded area features (i.e. create a box) when exporting extruded areas to 3D Shapefiles. The sides and bottom of the extruded area will be included in a multi-part polygon with the top of the 3D area.

Simple ASCII/CSV/XYZI Parameters

- **COORD_DELIM** specifies the delimiter between coordinates in coordinate lines
  - **COMMA** - coordinates are separated by commas
  - **SEMICOLON** - coordinates are separated by semicolons
  - **SPACE** - coordinates are separated by space characters
  - **TAB** - coordinates are separated by tab characters
- **COORD_ORDER** - specifies the order of the coordinates in coordinate lines. The following values are sup- ported:
  - **X_FIRST** - x coordinates (i.e. easting or longitude) come first, followed by y coordinates (i.e. north- ing or latitude) (default)
  - **Y_FIRST** - y coordinates (i.e. northing or latitude) come first, followed by x coordinates (i.e. easting or longitude)
  - **WKT** - coordinate string in WKT (well-known-text format). This allows single line representations of areas, lines, and points.
- **FEATURE_SEP** (SIMPLE_ASCII only) - specifies whether or not to separate vector features with a blank line

- **NONE** - do not separate vector features
- **BLANK_LINE** - separate vector features with a blank line
- *Any other text*. Use the escape sequence **\n** to specify that you want to insert a line break. For example, to separate features with a blank line, then a line with the text "NEW FEATURE", then another blank line, use FEATURE_SEP="\nNEW FEATURE\n".

- **PRECISION** specifies the number of digits to include beyond the decimal point for the stored values. For example use PRECISION=6 to store values like XXXXXXX.XXXXXX.

- **SAVE_DIST_Z_FILE** (SIMPLE_ASCII only) - specifies that the output file should be a distance/Z file. Use SAVE_DIST_Z_FILE=**YES** to enable this option.

- **USE_3D_DIST** (SIMPLE_ASCII only) - specifies if exporting a distance/Z file that the distances are the 3D along-ground distance and not just distances on a flat ellipsoid. Add USE_3D_DIST=**YES** to enable.

- **COORD_OFFSET** (SIMPLE_ASCII only) - specifies the offset to apply to any coordinates written to the file. This offset will be added to each coordinate written to the file. The offset should be specified as a *comma-delimited list of the X, Y, and Z offsets*, such as COORD_OFFSET=100000.0,200000.0,0.0

- **COORD_SCALE** (SIMPLE_ASCII only) - specifies the scale factors to apply to any coordinates written to the file. Each coordinate will be multiplied by these scale factor before being written to the file. The scale factors should be specified as a *comma-delimited list of the X, Y, and Z scale factors*, such as COORD_SCALE-E=0.1,0.1,1.0

- **EXPORT_ELEV** - specifies whether or not a elevation value should be generated for each vertex. A value of EXPORT_ELEV=**YES** will cause elevations to be generated. If the option is not specified, elevation values will be generated.

- **EXPORT_ATTRS**- specifies whether or not feature attributes should be written to the text file just before the coordinates. Use EXPORT_ATTRS=**YES** to enable export of the feature attributes. If the option is not specified, attributes will be exported. If you don't want to export style attribute with the feature, use EXPORT_ATTRS=**NO_STYLE** to get just the associated attributes and name of the feature in the attribute list.

- **ADD_LAT_LON** (CSV only) - specifies that lat/lon columns should be added to a CSV export. Use ADD_LAT_LON=**YES** to enable.

- **EXPORT_HEADER** (CSV only) - specifies whether or not the header line for CSV files should be written. Enabled by default, use EXPORT_HEADER=**NO** to disable.

- **USE_COMMA_FOR_DECIMAL** (CSV only) - specifies whether the European style of using a comma for a decimal should be used rather than using a period.

- **POINTS_ONLY** (CSV only) - specifies whether loaded area and line features should be included in a CSV export as WKT coordinate strings. The **default** is to only export points, so add POINTS_ONLY=**NO** to enable area and line export.

- **EXPORT_ECEF** (SIMPLE_ASCII only) - specifies that the export should use ECEF (earth-centered earth-fixed, or geocentric) coordinates.

DXF/DWG Parameters

- **EXPORT_DWG_LABELS** (DWG only) and **EXPORT_DXF_LABELS** (DXF only) - controls how object labels are exported to DWG/DXF files. The following values are supported:
  - **YES** or **ATTRS** - export labels as attributes tied to each feature
  - **NO** or **NONE** - don't export labels
  - **FEATURE_LAYER_POINTS** - export labels as point features with the layer matching the feature they are associated with
  - **LABEL_POINTS** - export labels as point features in a separate label layer

  Typically you want to set this to YES, unless you are working with a software package that cannot handle DWG/DXF files with attributes. Leaving out this parameter will cause feature labels to be discarded on export.

- **LAYER_ATTR** - specifies the attribute value to use from each feature for the layer name in the output file. The **default** is to use the feature description. See special Attribute Name parameter details for recognized values.

- **VERSION** (DWG only) - specifies the version of DWG to export. The following values are supported:
  - **R12**
  - **R13**
  - **R14**
  - **R15** (Autocad 2000) [Default]
  - **R18** (Autocad 2004)

- **R21** (Autocad 2007)
- **R24** (Autocad 2010)
- **R27** (Autocad 2013)

- **DWG_TEXT_SIZE** (DWG only) - specifies the multiplier value to apply to text sizes when exporting DWG file. Use this to control how large text is in the exported file.

- **DXF_TEXT_SIZE** (DXF only) - specifies the multiplier value to apply to text sizes when exporting DXF file. Use this to control how large text is in the exported file.

- **ALLOW_LONG_LABELS** (DXF only) - specifies that labels over 31 characters in the DXF file. Add ALLOW_LONG_LABELS=**YES** to enable this.

- **EXPORT_ELEV** - specifies whether or not a elevation value should be generated for each vertex. A value of EXPORT_ELEV=**YES** will cause elevations to be generated. If the option is not specified, elevation values will be generated.

- **EXPORT_ATTRS** - specifies whether or not feature attributes should be written to the file. Use EXPORT_ATTRS=**YES** to enable export of the feature attributes. If the option is not specified, attributes will not be exported.

Exporting Vector Files to a Spatial Database

- **TYPE** - *File-Based Spatial Databases* (Using these TYPE values requires that the FILENAME parameter also be specified to identify the spatial database to be used.)
    - **SPATIALITE** - Spatialite/SQLite
    - **FILE_GDB** - Esri File Geodatabase
    - **ESRI_PGEO** - Esri Personal Geodatabase

- **TYPE** - *Connection-Based Spatial Databases* (Using these type values requires that the SDB_CONNECTION_NAME parameter also be specified to identify the connection to be used.)
    - **ESRI_ARCSDE** - Esri ArcSDE Geodatabase
    - **MSSQLSERVER** - Micorsoft SQL Server Spatial
    - **MYSQL** - MySQL Spatial
    - **POSTGIS** - PostGIS/PostgreSQL
    - **ORACLE** - Oracle Spatial Database

- **SHAPE_TYPE** - When exporting to a spatial database, the SHAPE_TYPE parameter is required, and must contain only one shape type. Each exported table will contain a single geometry type.

- **FILENAME** - When exporting to a file-based spatial database, use this parameter to specify the full path to the database file. If exporting to an Esri File Geodatabase, this parameter must contain the directory containing the geodatabase (typically ends in ".gdb" even though it is a directory).

- **SDB_CONNECTION_NAME** - The name of the connection to be used to access a connection-based spatial database. Connections can be defined in the script using a DEFINE_SDB_CONNECTION command, or by using the Connection Manager in the Global Manager user interface. All of the connections defined in the Connection Manager are available for use in a script.

- **SDB_TABLE_NAME** - Each EXPORT_VECTOR command will output a single database table. Use this parameter to specify the name of the database table where the output will be stored. If the export uses gridding or splits the output by attribute, then multiple tables will be exported. The name specified here will be the base name, and will be modified for each output table based on the gridding and split-by-attribute parameters. This is similar to how the gridding process works when exporting a SHAPEFILE.

Polish MP Parameters

- **MAP_NAME** - specifies the name to use for the map. Typically defaults to the filename if not specified.

- **TEMPLATE_FILENAME** (POLISH_MP only) - specifies the full path and filename for another MP file to use for the settings for the new MP file being exported.

- **MP_EXPORT_TEMPLATE_FILES** (POLISH_MP only) - if a TEMPLATE_FILENAME value is provided, this controls whether or not the [FILES] section(s) from the template file will be copied to the new file.

- **MP_COPY_ENTIRE_TEMPLATE** (POLISH_MP only) - specifies that the entire contents of a specified template file should be copied to the new file rather than just the header portion of the template file.

- **MP_IMAGE_ID** (POLISH_MP only) - specifies the image ID value that should be stored in the resultant .mp file. If you don't specify this value or you specify a value of 0 a new value that has not been used before will automatically be generated.

- **EXPORT_ATTRS** specifies whether or not feature attributes should be written to the file. Use EXPORT_ATTRS=**YES** to enable export of the feature attributes. If the option is not specified, attributes will not be exported.

DGN Parameters

- **GEN_3D_FEATURES -** specifies that 3D line and point objects should be created in the exported file. Set this to **YES** to cause the the 3D features to be generated. Leaving out this parameter or setting it to anything but YES results in the normal 2D objects. The elevation stored for each vertex/point will be the first of the following that is available:
  - The elevation associated with the vertex/point in question.
  - The elevation associated with the entire area/line/point being exported. For example, the elevation of a contour line or spot elevation.
  - The first elevation obtained by searching the loaded elevation layers at the position of the vertex/point.

  A value of 0.0 will be used if no elevation could be obtained via any of the prior methods.

- **DGN_UNIT_RESOLUTION** - specifies the unit resolution to use in an exported DGN file. The **default** is **10000.0**.

- **DGN_GLOBAL_ORIGIN_LL** - specifies whether the global origin of the exported DGN file should be set to the lower left of the design plane rather than at the center of the design plane. Use DGN_GLOBAL_ORIGIN_LL=**YES** to move the global origin to the lower left.

- **DGN_REPLACE_DARK_COLORS** - specifies whether the color of dark lines should automatically be replaced with white on export to make them more visible on a dark background. Use DGN_REPLACE_DARK_COLORS=**YES** to enable this option.

- **EXPORT_ATTRS** - specifies whether or not feature attributes should be written to the DGN file as tags. Use EXPORT_ATTRS=**YES** to enable export of the feature attributes. If the option is not specified, attributes will be exported. If you don't want to export style attribute with the feature, use EXPORT_ATTRS=**NO_STYLE** to get just the associated attributes and name of the feature in the attribute list.

KML/KMZ Parameters

- **KML_AREA_DISPLAY_ABOVE_TERRAIN** (KML only) - specifies that area features with associated elevation value should be displayed at height above the terrain surface in Google Earth. Use KML_AREA_DISPLAY_ABOVE_TERRAIN=**YES** to enable.

- **KML_AREA_ELEVS_RELATIVE** (KML only) - specifies that the elevation values associated with 3D area features are relative to the terrain surface rather than relative to sea level. Use KML_AREA_ELEVS_RELATIVE=**YES** to enable.

- **KML_AREA_EXTRUDE** (KML only) - specifies that 3D area features displayed in Google Earth should be extruded from the surface to create volumetric objects like buildings. Use KML_AREA_EXTRUDE=**YES** to enable.

- **KML_AREA_FAKE_HEIGHTS** (KML only) - specifies that fake elevation values should be assigned to area features exported to ensure that the draw order remains correct in Google Earth. This may be necessary to keep overlapping area features drawing correctly. Use KML_AREA_FAKE_HEIGHTS=**YES** to enable.

- **KML_AREA_TRANSLUCENCY** (KML only) - specifies how see-through filled area features will be in the generated KML file. The values should range from **1** *to* **100** and represent a opacity percentage, with 100 being completely opaque and 1 being almost completely transparent. The **default** value is KML_AREA_TRANSLUCENCY=**75**.

- **KML_CHOP_FILLED_AREAS** (KML only) - specifies that filled areas with a large number of vertices (over 4096 currently) should be chopped up into smaller pieces to avoid rendering issues in Google Earth on some machines. Use KML_CHOP_FILLED_AREAS=**FALSE** to disable the chopping, which is automatically done for filled areas with no borders.

- **KML_FOLDER_ATTR** (KML only) - specifies the name of a feature attribute to use for the folder name in the generated KML file. By **default**, the export will check for a KML_FOLDER attribute with the name of a folder to use.

- **KML_HTML_DESC_TEXT** (KML only) - specifies a *HTML text string* describing what to use for the description for each feature exported to a KML file. To add a quote mark inside your description text, use two single quotes ('') rather than a double quote ("), as the latter would terminate the parameter value.

- **KML_LINE_DISPLAY_ABOVE_TERRAIN** (KML only) - specifies that LINE features with associated elevation value should be displayed at height above the terrain surface in Google Earth. Use KML_LINE_DISPLAY_ABOVE_TERRAIN=**YES** to enable.

- **KML_LINE_ELEVS_RELATIVE** (KML only) - specifies that the elevation values associated with 3D line features are relative to the terrain surface rather than relative to sea level. Use KML_LINE_ELEVS_RELATIVE=**YES** to enable.

- **KML_POINT_DISPLAY_ABOVE_TERRAIN** (KML only) - specifies that point features with associated elevation value should be displayed at height above the terrain surface in Google Earth. Use KML_POINT_DISPLAY_ABOVE_TERRAIN=**YES** to enable.

- **KML_POINT_ELEVS_RELATIVE** (KML only) - specifies that the elevation values associated with 3D Point features are relative to the terrain surface rather than relative to sea level. Use KML_POINT_ELEVS_RELATIVE=**YES** to enable.

- **KML_POINT_EXTRUDE** (KML only) - specifies that 3D point features displayed in Google Earth should be extruded from the surface by drawing a thin line from the surface to the point. Use KML_POINT_EXTRUDE=**YES** to enable.

- **INC_LAYER_ATTR** - specifies whether or not displays labels should be exported for line and area features. Use INC_LAYER_ATTR=**YES** to enable or INC_LAYER_ATTR=**NO** to disable. This is **disabled by default**.

- **CODE_PAGE** - specifies the code page to use for the KML export. The **default** is **UTF-8**. Use the *code page number*, the *ISO-8859-? string*, or the text *UTF-8* (number 65001).

Lidar LAS/LAZ Fields

- **ELEV_UNITS** - specify elevation units to use in export
    - **FEET** - export in US feet
    - **METERS** - export in meters

- **LAS_VERSION** - specifies what version of LAS file to write out. This would be **1.1**, **1.2**, **1.3**, or **1.4**. If you don't specify a version, the lowest version that will support all of the provided options will be used (typically 1.1 or 1.2).

- **VERT_CS_CODE** - specifies the vertical coordinate system (i.e. vertical datum) to store in the LAS file to specify what the elevations are referenced to. Use the *EPSG code*, like 5103 for NAVD88. If you don't specify a value and the source files used all use the same known system, that will be used. Note that no vertical datum conversion is done, this is just to supply metadata.

- **VERT_CITATION** - specifies the text description to store in the Lidar LAS file for the vertical coordinate system for the elevations. If nothing is supplied the **default** one (if any) for the supplied VERT_CS_CODE will be used.

- **FILE_SOURCE_ID** - specifies a File Source ID numeric value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.

- **GLOBAL_ENCODING** - specifies a Global Encoding numeric value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.

- **SYSTEM_ID** - specifies a System ID value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used.

- **GEN_SOFTWARE** - specifies a Generating Software value to store in the exported LAS file header. If not specified and one of the input files is a LAS file with this value specified it will be used. Otherwise '**Global Mapper**' will be used.

- **INC_COLOR** - specifies that a color value should be included with each point sample from the loaded raster layers (or from the original points if they were Lidar points with a color value). Use INC_COLOR=**YES** to enable.

- **NO_PROJ_HEADER** - specifies whether or not the current projection should be written in the LAS header. Use NO_PROJ_HEADER=**YES** to cause the projection to not be written.

- **HEADER_OFFSET** - specifies a custom header offset to use in the LAS file rather than an automatically determined one. The offset should be specified as a *comma-delimited list of the X, Y, and Z offsets*, such as `HEADER_OFFSET="100000.0,200000.0,0.0"`. If you want the offset used for the original input file (if a Lidar point cloud), use HEADER_OFFSET="**KEEP_ORIG**".

- **HEADER_SCALE** - specifies a custom header scale factor to use in the LAS file rather than an automatically determined one. The offset should be specified as a *comma-delimited list of the X, Y, and Z scales*, such as `HEADER_OFFSET="0.01,0.01,0.001"` to specify precision of 100ths of units in X and Y and thousandths in Z. If you want the offset used for the original input file (if a Lidar point cloud), use HEADER_SCALE="**KEEP_ORIG**".

- **FLIGHT_DATE** - specifies the flight date to store in the exported LAS file header. This can be either the day of the current year (value 1 to 366) or a common date format, including month, day, and year. If not specified the current date will be used.

Lidar Point Filter Options

See also Lidar Point Filter Options

GPX Fields

- **EXPORT_AREAS** - specifies that area features should be exported to GPX files as track lines. Use EXPORT_AREAS=**YES** to enable.
- **EXPORT_ATTRS** - specifies whether or not all feature attributes for waypoints will be included as tags. Use EXPORT_ATTRS=**YES** to enable export.
- **EXPORT_DESC** - specifies that <desc> tags will be exported for waypoints. Use EXPORT_DESC=**NO** to disable.
- **EXPORT_ELEV** - specifies whether or not a <ele> (elevation) value should be generated for each way-points/trackpoing. Use EXPORT_ELEV=**NO** to disable.
- **EXPORT_SYM** - specifies that <sym> (symbol) tags will be exported for waypoints. Use EXPORT_SYM=**NO** to disable.
- **EXPORT_TIME** - specifies that <time> tags will be exported for waypoints and track points when available. Use EXPORT_TIME=**NO** to disable.

Other Formats Fields

- **QUAD_NAME** (DLGO only) - specifies the quadrangle name to store in the header of the DLG-O file. If not quadrangle name is specified, Global Mapper will attempt to automatically determine one based on the loaded data.
- **PRECISION** (SVG only) - specifies the number of digits to include beyond the decimal point for the stored values. For example use `PRECISION=6` to store values like XXXXXXX.XXXXXX.
- **CDF_MAJOR_CODE** (CDF only) - specifies the default major attribute code to use for features when exporting to a CDF format file (**default** is **32**).
- **CDF_MINOR_CODE** (CDF only) - specifies the default minor attribute code to use for features when exporting to a CDF format file (**default** is **45**).
- **CDF_USE_DEFAULT_CODE** (CDF only) - specifies that the default attribute code pair should be used for all features written to the CDF file and not just those for which no attribute code pair could be automatically determined based on the feature classification.
- **INC_ELEV_ATTR** (MAPINFO only) - specifies whether or not the elevation of a feature should be added as an attribute to the exported files. Use INC_ELEV_ATTR=YES to enable or INC_ELEV_ATTR=**NO** to disable. This is enabled by default.
- **INC_LAYER_ATTR** (MAPINFO only) - specifies whether or not the layer (description) of a feature should be added as an attribute to the exported file. Use INC_LAYER_ATTR=**YES** to enable or INC_LAYER_ATTR-R=NO to disable. This is enabled by default.
- **VERSION** (LOWRANCE_USR only) - specifies which version of USR file to create. Must be **3** or **4**. By **default** VERSION=3 is used.
- **SAVE_XY_AS_TENTHS** (SEGP1 only) - specifies that the X and Y values should be multiplied by 10 when saved to the SEGP1 file
- **SAVE_Z_AS_TENTHS** (SEGP1 only) - specifies that the Z values should be multiplied by 10 when saved to the SEGP1 file
- **ATTR_TO_DELETE** (OSM only) - provides an attribute to ignore/delete when exporting to OSM XML files. You can provide multiple of these parameters to ignore multiple attributes, like `ATTR_TO_DELETE="MP_TYPE" ATTR_TO_DELETE="RouteParam"`.
- **Y_UP** (STL, COLLADA, PLY, and OBJ only) - specifies that the STL file that is created will use a "Y-Up" orientation; that is, Y values in exported coordinates will represent latitudes and Z values will represent elevations.
- **NO_PROMPTING** (STL, COLLADA, PLY, and OBJ only) - specifies that the 3D export dialog is suppressed.
- **CREATE_BINARY** (STL only) - specifies that the STL file that is created will be a binary STL file rather than a (much larger) ASCII text STL file.

EXPORT_WEB

The EXPORT_WEB command exports all currently loaded data to a tiled web format. The following parameters are supported by the command.

- **FILENAME** - output file name. For Google Maps, Bing, and OSM, this is the name of the HTML file that will be used to display the tiles. For TMS, this is the name of the XML file that contains the tile info. For KML Raster, this is the name of the KML/KMZ file. For MBTiles, this is the name of the SQLite database, with an extension of ".mbtiles". For RMaps, this is the name of the SQLite database, with an extension of ".sqlite". In all of these cases, the EXPORT_WEB command will create a new output file.

- **EXPORT_LAYER** - filename or description of layer(s) to export. By **default** all compatible and exportable layers are exported. You can include multiple EXPORT_LAYER parameters if you have multiple masks to search. Wildcards (* and ?) are supported. Hidden layers are not considered.

- **TYPE** - type of vector file we're exporting to
    - **GOOGLE_MAPS** - Google Maps Tiles
    - **VIRTUAL_EARTH** - Bing/Virtual Earth Tiles
    - **KML_RASTER** - KML file with all data in a raster
    - **WORLDWIND** - World Wind Tiles
    - **TMS** - Tile Mapping Service tiles
    - **OSM** - Open Street Maps Tiles
    - **MBTILES** - MapBox MBTiles SQLite Database
    - **RMAPS** - RMaps SQLite Database

- **MAX_ZOOM_LEVEL**

    - highest zoom level for which tiles will be created.

    Valid zoom levels, and associated resolution using the default 256 x 256 pixel tiles are:

    - **0** or *not provided* - automatically choose a default zoom level to capture the full detail of the layer(s) being exported
    - **3** - 19568 meters/pixel
    - **4** - 9784 meters/pixel
    - **5** - 4892 meters/pixel
    - **6** - 2446 meters/pixel
    - **7** - 1223 meters/pixel
    - **8** - 611 meters/pixel
    - **9** - 306 meters/pixel
    - **10** - 153 meters/pixel
    - **11** - 76 meters/pixel
    - **12** - 38 meters/pixel
    - **13** - 19 meters/pixel
    - **14** - 9.6 meters/pixel
    - **15** - 4.8 meters/pixel
    - **16** - 2.4 meters/pixel
    - **17** - 1.19 meters/pixel
    - **18** - 0.60 meters/pixel
    - **19** - 0.30 meters/pixel
    - **20** - 0.15 meters/pixel
    - **21** - 0.07 meters/pixel
    - **22** - 0.04 meters/pixel
    - **23** - 0.02 meters/pixel

- **NUM_ZOOM_LEVELS** - contains the number of zoom levels to be created. The **default** is **5**.

- **MAP_NAME** - user-defined name for the map.

- **IMAGE_FORMAT** - is the format to be used for the images. Choices are "**PNG**", "**JPG**"and "**GMG**". The **default** is "PNG". To export terrain tiles using the Global Mapper Grid (GMG) format, use `IMAGE_FORMAT=T="GMG"`.

- **QUALITY** - For JPG format images specify the quality

- **BG_MAP_NAME** (Google Maps tiles only) - type of map to use as the background:
    - **ROADMAP** - Road map
    - **SATELLITE** - Satellite imagery
    - **HYBRID** - Combination of ROADMAP and SATELLITE
    - **TERRAIN** - Terrain map

- **TRANSLUCENCY** - amount of translucency for the tiles. The value must be between **0.0** and **1.0**. 1.0 is Opaque, and the image gets more translucent as the numbers get lower. 0.0 is treated the same as 1.0. The **default** is 1.0.
- **TILE_PATH** - contains the directory where the tiles should be stored. If this is not specified, the tiles will be written to the directory specified for the FILENAME parameter.
- **CUSTOM_TILE_FILENAME** - custom definition for tile filenames. Use variables *%z* for zoom, *%x* for column, and *%y* for row. For example, use "*%z\\prefix_%y_%x.png*" to create one folder per zoom level. The tiles will be created in the path under the HTML filename path.
- **TILE_SIZE** - use this to override the default tile size for the selected TYPE. This value specifies the size of each tile in the resulting tile set. For example, using `TILE_SIZE=1024` will result in tiles of size 1024x1024 being created rather than the default (typically 256x256).
- **MBT_DESCRIPTION** (MBTiles only) - map description that will be added to the MBTiles metadata .
- **MBT_MAP_TYPE** (MBTiles only)- indicates the type of map . Valid values are (**default** is "overlay"):
  - **BASEMAP** - This map will be the base map.
  - **OVERLAY** - This map will overlay another map.
- **SPATIAL_RES** - specifies spatial resolution. Defaults to the minimum spatial resolution of the two layers if not specified. Should be formatted as *x_resolution,y_resolution*. The units are the units of the current global projection. For example, if UTM was the current global projection and you wanted to export at 30 meter spacing, the parameter/ value pair would look like `SPATIAL_RES=30.0,30.0`. You can also specify as a percentage of the default resolution by adding a percent. For example to get half the detail your double the spatial resolution value, so you would use `SPATIAL_RES="200%,200%"`.
- **SPATIAL_RES_METERS** - specifies spatial resolution to use in meters. The value in meters will automatically be converted to the current view/ export projection units. For example, to do an export at 2.0 meter spacing (or as close as you can get to that in the current units), use `SPATIAL_RES_METERS=2.0`, or to do an export at 1.0 meters in X by 1.5 meters in Y, use `SPATIAL_RES_METERS="1.0,1.5"`.
- **PIXEL_SIZE** - specifies the desired size in pixels of your export. Use this instead of SPATIAL_RES if you know exactly how many pixels in size your export should be. The format is PIXEL_SIZE="*widthxheight*". For example, to make your export have dimensions of 1024 pixels wide by 768 pixels tall, use `PIXEL_SIZE‐E="1024x768"`.

The following parameters accept boolean values ("YES" or "NO") to turn on or off the associated option (the default is no).

- **WEB_NO_TRANSPARENCY** - Do not use transparent background pixels
- **WEB_HIDE_PROGRESS** - Hide the progress bar windows
- **WEB_ADD_SCALE_BAR** (Google Maps Only) - Add a scale bar to the map
- **WEB_ADD_MAP_TYPE_CONTROL** (Google Maps Only) - Add a map type control
- **WEB_ADD_OVERVIEW_MAP** (Google Maps Only) - Add an Overview Map
- **WEB_AUTO_GRID** (KML Raster only) - Create a default grid
- **WEB_SKIP_EMPTY_TILES** - Skip tiles that have no data
- **WEB_USE_JAVA_FILE_NAMES** (World Wind only) - Create names for World Wind Java
- **WEB_TRANSPARENT_TILES** - Make the image tiles transparent
- **WEB_SKIP_EXISTING_TILES** - Skip tiles that already exist (resuming export)
- **WEB_FILL_TO_TILE_BOUNDS** - Fill the tiles to the bounds
- **WEB_NO_HTML_FILE** - Generate tiles only, no HTML file
- **WEB_FORCE_PALETTE_PNG** - Force the PNG to have palette instead of RGB
- **WEB_FULL_TILES_ONLY** - Only export tiles that are fully covered
- **Specify Bounding Box for Operation**

- **Cropping to Polygons/Areas**

Shared Parameters

Cropping Operations to Polygons/Areas

Most commands that support cropping can crop to a polygon using the POLYGON_CROP_FILE and related parameters. With these you can specify to crop the operation to area features from some vector file or already loaded layers. See below for a detailed description of the parameters that are related to polygon cropping:

- **POLYGON_CROP_FILE** - specifies the full path and filename or loaded layer description of a vector file/-loaded layer containing one or more polygon features to which the operation should be cropped. If multiple polygons are found in the specified file the polygon which has the largest intersection with the data to be combined will be used as the crop polygon (see POLYGON_CROP_USE_ALL or POLYGON_CROP_USE_EACH for exceptions).

- **POLYGON_CROP_FILE_PROJ** - specifies the projection to use for the POLYGON_CROP_FILE. Use if the file doesn't have an associated projection file. See special Projection Specification for instructions.

- **POLYGON_CROP_NAME** - specifies the name of a polygon shape previously defined using the DEFINE_SHAPE command to which the export should be cropped. The coordinates in the shape need to have been provided in whatever projection the new terrain layer will be in (i.e. the current projection). If you want to crop to any area features that are selected with the Digitizer Tool in the user interface rather than a defined polygon, use POLYGON_CROP_NAME="**SELECTED**".

- **POLYGON_CROP_USE_ALL** - specifies that if a POLYGON_CROP_FILE is specified that contains multiple polygons, the operation will be cropped to all polygons in that file rather than just the best-fit polygon.

- **POLYGON_CROP_USE_EACH** - specifies that if a POLYGON_CROP_FILE is specified that contains multiple polygons, the operation will generate a separate export for each polygons in that file rather than just the best-fit polygon. See the POLYGON_CROP_BBOX_ONLY and POLYGON_CROP_NAME_ATTR options for naming and other options when using this parameter. Use POLYGON_CROP_USE_EACH=**YES** to enable.

- **POLYGON_CROP_EXCLUDE** - specifies that the crop areas are actually regions to exclude from the export rather than include. If you add POLYGON_CROP_EXCLUDE=**YES** to the command the results will contain everything outside the crop areas but within the full export bounds.

- **POLYGON_CROP_COMBINE_DUPS** - specifies that if a POLYGON_CROP_FILE is specified that contains multiple polygons and POLYGON_CROP_USE_EACH is set, whether or not polygons with duplicate values for the attribute used for filenaming will be combined into a single export or split into separate exports. The default is POLYGON_CROP_COMBINE_DUPS=**YES**.

- **POLYGON_CROP_BBOX_ONLY** - specifies that if the POLYGON_CROP_USE_EACH parameter is specified that each export should just be cropped to the bounding box of each polygon rather than the actual boundary of the polygon. Use POLYGON_CROP_BBOX_ONLY=**YES** to enable only cropping to the bounding box.

- **POLYGON_CROP_GRID_ONLY** - specifies that any crop polygon(s) are used only to determine which tile/grid cells will be exported. For each tile/grid cell that intersects a crop polygon, the full tile/grid cell will be exported. Use POLYGON_CROP_GRID_ONLY=**YES** to enable.

- **POLYGON_CROP_INT_DATA_BOUNDS** - control how the bounding box for the export is determined if a crop polygon is specified. If this option is disabled, the export bounds will be the bounds of the crop area(s), even if they extend outside the data bounds, unless you explicitly specified a bounding box in the other parameters (i.e. LAYER_BOUNDS, GLOBAL_BOUNDS, etc.). If the option is enabled, the export bounds will be the intersection of the crop area bounds and the non-cropped bounding box (i.e. full data bounds or specified bounds in other parameter). This parameter is enabled by default, use POLYGON_CROP_INT_DATA_BOUNDS=**NO** to disable.

- **POLYGON_CROP_NAME_ATTR** - used to control the filenames generated when cropping to multiple polygons using the POLYGON_CROP_USE_EACH parameter. See special [Attribute Name](#) parameter details. This value will be appended to any filename specified in the EXPORT FILENAME parameter. If no value is provided, the exported files will be sequentially numbered.

- **POLYGON_CROP_FOLDER_ATTR** - used to control the filenames generated when cropping to multiple polygons using the POLYGON_CROP_USE_EACH parameter. See special [Attribute Name](#) parameter details.

- **POLYGON_CROP_FILENAME_SUFFIX** - specifies the text to insert in the output filename just before the file extension when cropping to polygons. For example, if you add `POLYGON_CROP_FILENAME_SUFFIX="_out"`, the original FILENAME was "my_file_.dem" and the value from the polygon was "A1", you would get an output filename of "my_file_A1_out.dem".

- **POLYGON_CROP_COMPARE_STR** - specifies a compare string to use to filter out the areas in the polygon crop file. See the COMPARE_STR parameter for the [EDIT_VECTOR](#) script command for details.

EXAMPLE

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
EXPORT_RASTER FILENAME="C:\TEMP\" SPATIAL_RES="1, 1" TYPE=GEOTIFF\
POLYGON_CROP_FILE="ne_10m_admin_0_countries.shp" POLYGON_CROP_USE_EACH=YES POLYGON_CROP_NAME_
ATTR="<Feature Name>" POLYGON_CROP_FILENAME_SUFFIX="_crop.tiff"
```

Gridding/Tiling Operations into Smaller Chunks

The following parameters apply to export and other commands that support breaking the operation up into regular tiles of some size rather than generating just a single output:

- **GRID_TYPE_CELL_SIZE** - specifies that the export should be gridded into multiple tiles with each tile having the given size in the export projection. The value should be specified as *cell width,cell height*. For example, if you are exporting in a metric projection and want to tile the export into cells that are 10,000 meters wide by 5,000 meters tall, you would use `GRID_TYPE_CELL_SIZE="10000.0,5000.0"`.

- **GRID_TYPE_PIXEL_SIZE** - specifies that the export should be gridded into multiple tiles with each tile having the given size in pixels/samples. The value should be specified as *cell pixel width,cell pixel height*. For example, if you want to tile the export into cells that are 800 pixels wide by 600 pixels tall, you would use `GRID_TYPE_PIXEL_SIZE="800,600"`.

- **GRID_TYPE_PIXEL_SIZE_MAX** - specifies that the export should be gridded into multiple tiles with each tile having the given maximum size in pixels/samples. This works by calculating the number of rows and columns needed to hold cells of the given pixel dimensions, then shrinking those pixel dimensions down so that the tiles exactly cover the exported bounds. The value should be specified as *cell pixel width,-cell pixel height*. For example, if you want to tile the export into cells that are at most 800 pixels wide by 600 pixels tall, you would use `GRID_TYPE_PIXEL_SIZE_MAX="800,600"`.

- **GRID_TYPE_ROWS_COLS** - specifies that the export should be gridded into multiple tiles with a given number of rows and columns of tiles. The value should be specified as *number of rows,number of columns*. For example, if you want to tile the export into a grid of 8 rows each 4 tiles across, you would use `GRID_TYPE_ROWS_COLS="8,4"`.

- **GRID_OVERLAP** - specifies how much to overlap tiles when gridding an export into multiple tiles. This is a percentage value from 0 to 100 and only applies when on of the GRID_TYPE_* parameters is used. For example, to make your grid tiles overlap by 5% of the grid tile size, use `GRID_OVERLAP="5.0"`. The **default** value is **0.0**, meaning that the tiles do not overlap.

- **GRID_OVERLAP_NUM_PIXELS** - indicates whether or not the value specified in GRID_OVERLAP should be interpreted as a number of pixels instead of a percentage. Use GRID_OVERLAP_NUM_PIXELS=**YES** to specify that the value specified in the GRID_OVERLAP parameter should be interpreted as a number of pixels. Use GRID_OVERLAP_NUM_PIXELS=**NO** or omit this parameter to specify that the value is a percentage.

- **GRID_KEEP_CELL_SIZE** - specifies that the size of the grid cells should be maintained over sample spacing. This means that if you specify a grid of 4 rows and 5 columns, each grid cell will be exactly 25% of the total export height and 20% of the total export width. The sample spacing may be slightly smaller than what is specified in order to achieve this. By **default**, the sample spacing is exactly maintained and each

grid cell may be slightly larger than specified to maintain an integer number of exported cells. Use GRID_ KEEP_CELL_SIZE=**YES** to enable.

- **GRID_NAMING** - specifies how to name tiles when gridding an export into multiple tiles. The value should be **SEQUENTIAL** for sequential numeric naming starting at 1, **SEPARATE** for separate prefix appending by row and column, or **SEPARATE_COLS_FIRST** for separate prefix appending by columns and rows. For the SEPARATE options, use the GRID_NAMING_COLS and GRID_NAMING_ROWS parameters to specify the details of how to name the rows and columns. The value will be appended to FILENAME specified in the EXPORT command. If no GRID_NAMING parameter is supplied, the last selected grid naming options selected in the user interface will be used.

- **GRID_NAMING_COLS** - specifies how to name the column portion of grid cell names when using the GRID_NAMING=SEPARATE or GRID_NAMING=SEPARATE_COLS_FIRST parameter. The value of this field is a *comma-delimited list* with the following field values:
  - *Naming type*. Can have the following values:
    - **NUM** - name using numbers in ascending order
    - **NUM_REVERSE** - name using numbers in descending order
    - **ALPHA** - name using letters in ascending order
    - **ALPHA_REVERSE** - name using letters in descending order
  - *Starting value* for numbering or lettering (i.e. '1', or 'A'). If the naming type is numeric you can also specify *%left%* or *%right%* as the starting value to use the left or right coordinate of the cell bounding box. For row naming you can use *%top%* or *%bottom%*.
  - *Prefix string* to use before the numeric or alphabetic value.
  - *Step value* for numeric naming (default is '**1**')

  You can leave values blank if they don't apply or you want to use the default. As an example, to do numeric naming starting at the number 100, increasing by 10 each time with a prefix of DEM, you would use `GRID_ NAMING_COLS="NUM,100,DEM,10"`.

- **GRID_NAMING_ROWS** - specifies how to name the row portion of grid cell names when using the GRID_ NAMING=SEPARATE parameter. See the documentation for the GRID_NAMING_COLS parameter above for details on the format.

- **GRID_NAMING_PREPEND_ZEROES** - specifies whether or not to prepend zeroes to the start of grid column/row names. Use GRID_NAMING_PREPEND_ZEROES=NO to disable the prepending of zeroes.

- **GRID_NAMING_SEPARATOR** - specifies the separator string to use between pieces of a grid name. The default is an underscore (**_**).

- **GRID_CREATE_FOLDERS** - specifies that a separate folder should be generated for each row (or column if GRID_NAMING=SEPARATE_COLS_FIRST is specified) of the export rather than placing every output file in the same folder.

- Tiling to Polygons/Areas - If you would like to tile your export to a series of polygons/areas, use the POLYGON_CROP_FILE and POLYGON_CROP_USE_EACH parameters. Click the link for details.

Built-in Variables

For any command that breaks up an operation in to multiple pieces using gridding, you can use one of the special character sequences below in a parameter of the command to use a piece of information about the grid cell being exported in the export (i.e. parts of grid cell filename). The examples of what the values will be based on a current grid filename of 'C:\path\to\my\data\my_file_A1.dem' are listed:

- *%TILE_DIR%* - full path to current file (value is 'C:\path\to\my\data\')

- *%TILE_FNAME_W_DIR%* - full path and filename of current file (value is 'C:\path\to\my\data\my_file_A1.- dem')

- *%TILE_FNAME%* - filename of current file (value is 'my_file_A1.dem')

- *%TILE_FNAME_WO_EXT%* - filename of current file without extension (value is 'my_file_A1')

Specify Bounds for Operation

The following parameters apply to export and other commands that support providing a bounding box within which to perform the operation. You can also use the Polygon Crop Parameters for some operations rather than providing a bounding box. In most cases if no bounding box is provided the combined bounding box of all layers used for the operation will be used:

- **GLOBAL_BOUNDS** - specifies the combine bounds in units of the current global projection. There should be 4 values in a comma-delimited list following the parameter name. The values should be in order of *minimum x, minimum y, maximum x, maximum y*.

- **GLOBAL_BOUNDS_SIZE** - specifies the combine bounds in units of the current global projection. There should be 4 values in a comma-delimited list following the parameter name. The values should be in order of *minimum x, minimum y, width in x, width in y*.

- **LAT_LON_BOUNDS** - specifies the combine bounds in latitude/longitude degrees. There should be 4 values in a comma-delimited list following the parameter name. The values should be in order of *west-most longitude, southern-most latitude, eastern-most longitude, northern-most latitude*.

- **LAYER_BOUNDS** - specifies that the operation should use the bounds of the loaded layer(s) with the given filename. For example, to export to the bounds of the file "c:\test.tif", you would use `LAYER_BOUNDS-S="c:\test.tif"`. Keep in mind that the file must be currently loaded.

- **LAYER_BOUNDS_EXPAND** - specifies that the operation should expand the used LAYER_BOUNDS bounding box by some amount. The amount to expand the bounding rectangle by should be specified in the current global projection. For example, if you have a UTM/meters projection active and want to expand the bounds retrieved from the LAYER_BOUNDS parameter by 100 meters on the left and right, and 50 meters on the top and bottom, you could use `LAYER_BOUNDS_EXPAND="100.0,50.0"`. You can also specify a single value to apply to all 4 sides, or supply 4 separate values in the order *left,top,right,bottom*.

- **SNAP_BOUNDS_TO_MULTIPLE** - specifies that the top-left corner of the bounding box for the operation should be snapped to a multiple of the given value. For example, using `SNAP_BOUNDS_TO_MULTIPLE=1` will snap the top-left corner to the nearest whole number. The values will always go smaller for X/easting/longitude and larger to Y/northing/latitude so you always get at least what is requested.

- **SNAP_BOUNDS_TO_SPACING** - specifies that the top-left corner of the bounding box for the operation should be snapped to a multiple of the resolution of the operation. For example, if you are exporting at 5 meter spacing, the top left corner will be snapped to the nearest multiple of 5. Use SNAP_BOUNDS_TO_SPACING=**YES** to enable or SNAP_BOUNDS_TO_SPACING=**NO** to disable. If not provided, the global setting for snapping exports to the nearest sample spacing boundary from the Advanced section of the General tab of the Configuration dialog will be used.

- **USE_EXACT_BOUNDS** - specifies that the exact bounds that were defined in the command should be used. Generally, when the bounds specified in a command are not the same as the data bounds, the command uses the intersection between the two. When USE_EXACT_BOUNDS=**YES** is specified, the command will use the bounds as specified, instead of the intersection.

Batch Mode Operation

You can run a Global Mapper script file automatically by passing it on the command line to the Global Mapper .exe file. The script file will be run with no user interface displayed and Global Mapper will immediately exit when the script file completes processing. This allows you to easily run Global Mapper scripts from another application or from a DOS batch file.

> Note that your script files need to have an extension of .gms for this to work.

Batch variables

When running in batch mode you can define variables on the command line so they will be available when running the script. You provide pairs of tokens on the command line, after the file name. Each pair must look like:

-<var name> <var value>

or

/<var name> <var value>

Example command line:

```
global_mapper.exe "c:\temp\myscript.gms" -var1 01 -var2 33
```

This defines two variables that can be used in the script: var1=01 and var2=33. See the DEFINE_VAR command for details on how to use variables.

Batch options

/showprogress - if present, instructs Global Mapper to display progress bars while processing a script. This parameter only has an effect when running a script from the command line.

Example command line:

```
"C:\Program Files\GlobalMapper17_64bit\global_mapper.exe" "C:\Scripts\export.gms" /showprogress
```

Sample Scripts

Crop, Merge, and Reproject 4 USGS DRGs into new GeoTIFF and JPEG files

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
UNLOAD_ALL
// Import the four 24K DRGs that we want to merge. We use the CLIP_COLLAR option
// to indicate that we want the collar to be automatically removed from the
// DRGs when they are imported.
IMPORT FILENAME="C:\DATA\DRG\KANSAS CITY\O39094B2.TIF" \
 TYPE=AUTO ANTI_ALIAS=NO AUTO_CONTRAST=NO CLIP_COLLAR=AUTO TEXTURE_MAP=NO
IMPORT FILENAME="C:\DATA\DRG\KANSAS CITY\O39094A1.TIF" \
 TYPE=AUTO ANTI_ALIAS=NO AUTO_CONTRAST=NO CLIP_COLLAR=AUTO TEXTURE_MAP=NO
IMPORT FILENAME="C:\DATA\DRG\KANSAS CITY\O39094A2.TIF" \
 TYPE=AUTO ANTI_ALIAS=NO AUTO_CONTRAST=NO CLIP_COLLAR=AUTO TEXTURE_MAP=NO
IMPORT FILENAME="C:\DATA\DRG\KANSAS CITY\O39094B1.TIF" \
 TYPE=AUTO ANTI_ALIAS=NO AUTO_CONTRAST=NO CLIP_COLLAR=AUTO TEXTURE_MAP=NO
// Load a projection file to set the global projection to geographic (lat/lon)
// arc degrees with a datum of NAD83.
LOAD_PROJECTION FILENAME="C:\DATA\PRJ Files\geo_degrees_nad83.prj"
// Use the EXPORT_RASTER command to generate a new 8-bit per pixel GeoTIFF file
EXPORT_RASTER FILENAME="C:\DATA\EXPORTED DATA\merged_drg_8bpp.tif" \
 TYPE=GEOTIFF PALETTE=OPTIMIZED
// Now, use the EXPORT_RASTER command to generate a grayscale GeoTIFF file. Lets
// also create a world file for this one
EXPORT_RASTER FILENAME="C:\DATA\EXPORTED DATA\merged_drg_gray.tif" \
 TYPE=GEOTIFF PALETTE=GRAYSCALE GEN_WORLD_FILE=YES
// Create a JPEG file using the EXPORT_RASTER command. Also create a world file
// and a projection file to make it easier to load in other places.
EXPORT_RASTER FILENAME="C:\DATA\EXPORTED DATA\merged_drg.jpg"\
 TYPE=JPEG GEN_WORLD_FILE=YES GEN_PRJ_FILE=YES
```

Generate Contours from all USGS DEMs in a Folder and Export them to DXF and Shape files

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
UNLOAD_ALL
// Loop over all DEM files in a folder and convert them
DIR_LOOP_START DIRECTORY="C:\DATA\SDTS_DEM\24K\" FILENAME_MASKS="*.DEM.STDS.TAR.GZ" RECURSE_
DIR=NO
// Import an archived SDTS DEM file. Global Mapper will automatically
// determine that this is an archived SDTS DEM file and load it
// correctly.
IMPORT FILENAME="%FNAME_W_DIR%" ANTI_ALIAS=YES
// Generate 50 ft contours from the loaded DEM data.
GENERATE_CONTOURS INTERVAL=50 ELEV_UNITS=FEET
// Export the contours to a new DXF file. The created file will have
// 3D polyline features for the contours.
EXPORT_VECTOR FILENAME="%DIR%%FNAME_WO_EXT%_CONTOURS.DXF" TYPE=DXF GEN_PRJ_FILE=YES
// Export the contours to a 3D shape file.
EXPORT_VECTOR FILENAME="%DIR%%FNAME_WO_EXT%_CONTOURS.SHP" TYPE=SHAPEFILE \
SHAPE_TYPE=LINES GEN_3D_LINES=YES GEN_PRJ_FILE=YES
// Unload the loaded data
UNLOAD_ALL
```

```
// End the loop
DIR_LOOP_END
```

Edit Vector Features Based on an Attribute and Display Label

```
GLOBAL_MAPPER_SCRIPT VERSION=1.00
// Import the file to modify
IMPORT FILENAME="C:\Temp\export test\tiger_wyandotte_sample.gmp"
// Assign the type "railroad" to all features with a CFCC attribute with a value of A41
// and a display label with '74' in it somewhere.
EDIT_VECTOR LINE_TYPE="RAILROAD" COMPARE_STR="CFCC=A41" COMPARE_STR="<Feature Name>=*74*"
// Assign the name "Burlington Northern Railroad" to all features with a CFCC attribute with a
value of A41
EDIT_VECTOR ATTR_VAL="<Feature Name>=Burlington Northern Railroad" COMPARE_STR="CFCC=A41"
```