HEXAGON

GeoCompressor 2020 Update 3

User Guide

Version 16.6.0 Update 3
11 February 2021

Contents

What's New

2020 Update 3

- Maintenance release
- Added resampling options Nearest Neighbour, Bilinear, Cubic, Cubic Spline, Lanczos, Average, Mode or Gauss when resizing data. #GC-1174
- ECW check is now more accurate when determining validity of files and prints better warnings. #EC-2444
- ECW check on Linux reports not enough disk space errors #GC-1194
- GeoCompressor cannot remove bands from the input image and change the cell size in same task #GC-1173
- If an existing JP2 file is opened for write, the file is not truncated and remains the same size as the original #GC-1195
- Error messages from GDAL are now captured in the processing log. #GC-1218
- Add confirmation dialog when stopping/deleting tasks from the UI. #GC-1235

2020 Update 2

- Maintenance release
- Optimised default profile for ECW version 3 files for faster decompression and lower storage requirements. #EC-2491
- Viewer crash on RGBA image when opacity channel selected in statistics. #GC-1131
- Make background color setting more consistent. Mosaic now honors background color setting and is applied to NULL block decode setting for ECW v3. #GC-426
- Added missing library dependencies for RHEL 8/CentOS 8 (new supported platform). #GC-1070
- Logfile now prints the tile size and compression metadata from GDAL. #GC-1133
- Block size for ECW files is now shown in Viewer properties page. #GC-1151
- Provide shapefile feature selector UI to region selection. #GC-1141

2020 Update 1

- Maintenance release
- PNG images with alpha would crash the compressor. #GC-1101
- GeoCompressor not recognizing the SRS information of some JP2 images. #GC-1100
- Viewer: Double clicking a file in the ECWP Browser that is not in the default service generates an error. #GC-1096
- NODATA value doesn't work for thematic input. #GC-1091
- License refresh button crashes UI when no license sources configured. #GC-1087
- Viewer: Cannot change log level. #GC-1084
- Deleting a task from the task scheduler results in corrupted reporting. #GC-1044
- Can't JP2 compress a 27-bit IMG (stored as 32-bit). #GC-849
- TIFF with rotation compressed to JP2 differs on Linux and Windows. #GC-794

- Cannot compress to JP2 with bit-depth other than 8 or 16. # GC-69

2020

- First release of GeoCompressor Viewer, a free to use general purpose viewer for ECW, JPEG2000 files designed to support rapid user display but also QA from Data Providers.
- The previously known “ERDAS APOLLO Utilities” have now been migrated as a new GeoCompressor Utilities component for ECWCheck, HeaderEditor and other supporting data preparation tools.
- GeoCompressor 2020 now enforces new core/thread restrictions based on detected license level. See licensing chapter for more details. #GC-969
- SOURCEDIR naming variable was erroneously applied in the wrong field. #GC-1059
- When compressing input data in EPSG:4326 the output projection was being lost due to mishandling of exponent GMLJP2 parsing. #GC-1074
- ECW v3 with stored histograms was incorrectly dropping the last bin value across all bands. #GC-1047
- Numerous improvements to unit and cell size persistence across platforms and workflows. This resolves issues such as output unit being lost or the wrong cell size being applied. #GC-1014, GC-998
- Log button is now enabled during compression jobs to make it simpler to monitor for any issues. #GC-1012
- Reworked the general handling of bitdepths to return more meaningful error states and support int16 or greater in more situations. There were some cases where previous versions were casting in16 to uint16 unexpectedly. #GC-1071, GC-1024, GC-995
- Deprecated “low memory handling” and now returns a hard error when insufficient memory is detected. #GC-949
- Enhanced the reliability in resolving input georeferencing to expected EPSG codes. #GC-1075
- Installation technology has been changed from the cross-platform Qt installer to platform specific, MSI (Windows) and RPM (Linux). #GC-872
- Resolved multithreading issue in reading GeoPackage Raster tables that caused random small white appears to appear in the output ECW. # GC-953
- General image compression throughput speeds can be expected to be 15% or higher on Intel® hardware.
- Variety of branding and other small user interface improvements
- Miscellaneous bug fixes and platform updates

2018 Update 3

- Added “-openoptions” parameter to command line compressor to pass parameters to GDAL input reader (refer to GDAL format driver documentation for available options for each format). #GC-800
- Calculating the default cell size in GUI results in incorrect values when map units are degrees. #GC-886
- Estimated time to completion always 0 for JPEG 2000 output. #GC-836
- When output extents are specified in XML the band list inputs are ignored. #GC-858
- Better reporting of license status in point compressor. #GC-851

2018 Update 2

- Fixed generation of GML in JP2 and GeoJP2 from GUI. #GC-840 / GC-841
- Add additional supported input types to the GUI when selecting via the file chooser. #GC-837

- Add estimated progress completion time to GUI. #GC-836
- Fixed crash when mosaicking images with different input band lists. #GC-796
- Miscellaneous UI bugs, clean-up and updated documentation.

2018 Update 1

- Fixed customer hang that could occur when mosaicking JPEG2000 input files. #GC-742 / 00025349
- Fixed inconsistent warning about thread count “exceeding CPU Core count”. #GC-740 / 00024548
- Clarified handling of input files where an unsupported GeoTransform is detected in the input. #GC-768 / 00026035
- Fixed UI errors when attempting to batch compress multiple images, or images without georeferencing. #GC-764, GC-765
- Null and data block counts are now correctly logged, where previously they were only printed to the console. #GC-762
- Numerous internal bugs were resolved to ensure binary consistency across Windows and Linux generated outputs. #GC-756, GC-422, GC-757
- Fixed crash caused by a custom bandlist order. It will now honour the band count and order specified when mosaicking. #GC-778

2018

- New workflow for creating multiple output files that are clipped to a polygon selected from a shape file. Users can create an output file for each state or county, where a shapefile containing the polygons defining those states or counties is specified, from either a single file or a virtual mosaic of many input files. Opacity channels are also generated from the selected polygons.
- Output scaling/resampling to new cell size
- Advanced customization options for JPEG 2000 file creation, including progression order, quality layers, SOP/EPH markers, and tile and precinct dimensions.
- Control over output SRS format for JPEG 2000 can now be set to be either GML in JPEG 2000 (v1.0) or GeoJP2.
- Ability to specify the input file band mappings in the GUI

2016

- Support for specifying a transparent colour and nodata values on input files
- Speed up when mosaicking large amounts of input imagery
- Add new feature to upload datasets to a Smart M.App Chest account for further processing on the cloud
- Clip output image to polygon bounding box when compressing
- AVX2 and FMA3 acceleration on supported CPUs

2015

- “ERDAS Image compressor” now known as “GeoCompressor”

    - Product now available for purchase under a subscription, tiered licensing scheme
- Point cloud conversion to the stream-able Hexagon Point Cloud (HPC) format has been added.
    - Windows only
- Partial region updates, or selective compression within an existing ECW v3 file is now supported.
- Image Mosaicking now defaults to the tile compression method for vastly improved performance.
- User interface has been redesigned.
- Many bug fixes, performance and general workflow improvements.
    - Image Compressor report now breaks down user time into read, write and reassembly time
    - Multi-band uint16 ECW v3 compression now uses per-band statistics to generate higher quality output
    - Region improvements to support multi-part features

2014.1

- Mosaicking now supports multiband input, multiband output. Previously only RGB/RGBA output was supported in 14.0.
- Mosaicking now supports greater than 8bit output for ECW v3 and JPEG2000 files.
- Parsing of mosaic projects with thousands of datasets now 10 times faster.
- Overriding of SRS information now possible in Mosaic XML as a new output option.
- Added support for MBTile, NITF and Terrashare RBD file input.
- 12-bit JPEG compressed file types now supported.
- Multi-part polygon regions now supported (e.g. polygons with holes).
- Further optimizations and bug fixes.

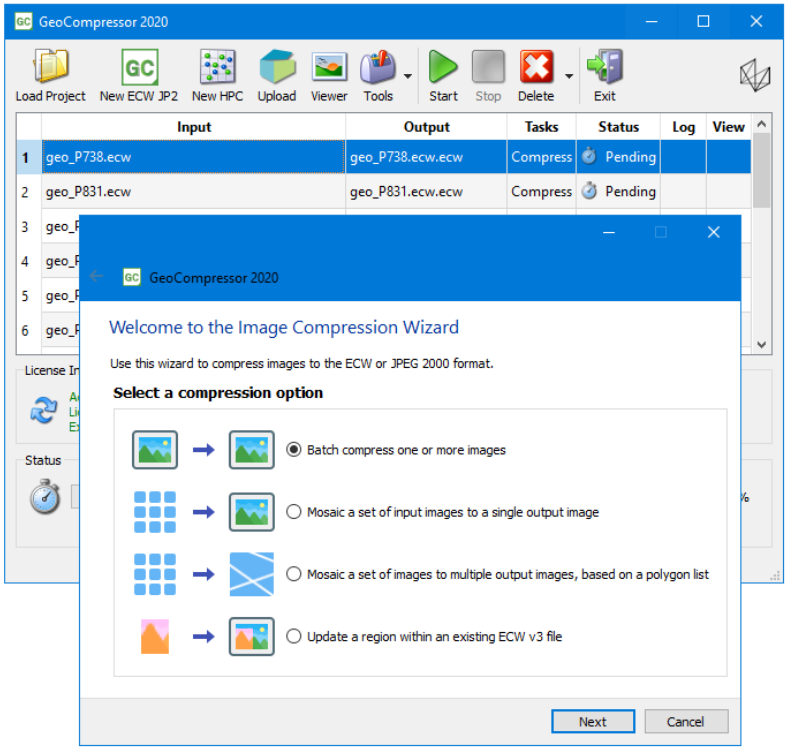
Overview

GeoCompressor is a high-performance geospatial image and point cloud compression application designed to simplify the creation of ECW, JPEG2000, and HPC formats. The application provides:

- Cross-platform, 64-bit support for both Windows and Linux with command-line and simple wizard user interfaces
- A stand-alone, decoupled tool to plugin to existing data processing workflows ideal for Data Providers
- A cost-effective solution that fills the void between Developers acquiring an ECWJP2 SDK license and a full ERDAS IMAGINE License
- The product's functional scope is intentionally narrow. The GeoCompressor does not:
  - Support 32-bit platforms
  - Offer advanced mosaicking functions
    - You cannot feather, blend, colour-balance, dodge, adjust seam-lines, preview output or perform many other tasks typically performed in a mosaic process
  - Reproject or warp imagery to different output coordinate systems
  - Support mosaicking LAS/LAZ Point clouds

GeoCompressor can be seen therefore as a transcoder only of input data. For other image and point-cloud processing tasks ERDAS IMAGINE remains the recommended tool for end-to-end processing and compression.

Figure 1 – Wizard compression interface



Architecture

GeoCompressor can be broken into input readers, compressor logic and output writers. The input readers ensure wide industry format support by leveraging the GDAL, libLAS, LASZip, and ERDAS ER Mapper libraries. Conceptually GeoCompressor comprises of three main utilities, *ImageCompressor, PointCompressor* and *MAppUploader.*

The application is thread-safe[1] and has been deployed on processing workstations up to 64-cores with excellent CPU scaling using the parallelized ECW compressor. Peak performance has been recorded in excess of 750MB/sec[2] encoding a 1000 gigapixel image and far exceeds the throughput of other third-party applications that implement the ERDAS ECWJP2 SDK due to optimizations implemented in the GeoCompressor architecture.

The provided user interface is a thin wrapper around several command-line executables, namely the ImageCompressor, PointCompressor and MappUploader executables. For intermediate to advanced users it is expected that most will interface with the command line tools to give more control integrating with existing workflows.

[1] Not all GDAL supported formats are Thread-safe. See FAQ.

[2] Compression throughput speeds vary and are heavily dependent on hardware and the input data format type and internal structure. This number is based on recording speed compressing an input test-pattern.

Supported Environments

OS = One of the operating systems is required.
DBS = One of the database server engines is required.
DBC = One of database client engines is required.
IB = One of the internet browsers is required.
TP = One of the third-party products is required.

R = Required
O = Optional
U = User must install

	GeoCompressor
Operating Systems	
Microsoft Windows® 8.1 & 10 (64-bit)	OS
Microsoft Windows® Server 2016 & 2019	OS
Red Hat® Enterprise Linux® / CentOS 8.x (64-bit)	OS
Red Hat® Enterprise Linux® / CentOS 7.x (64-bit)	OS

Note:

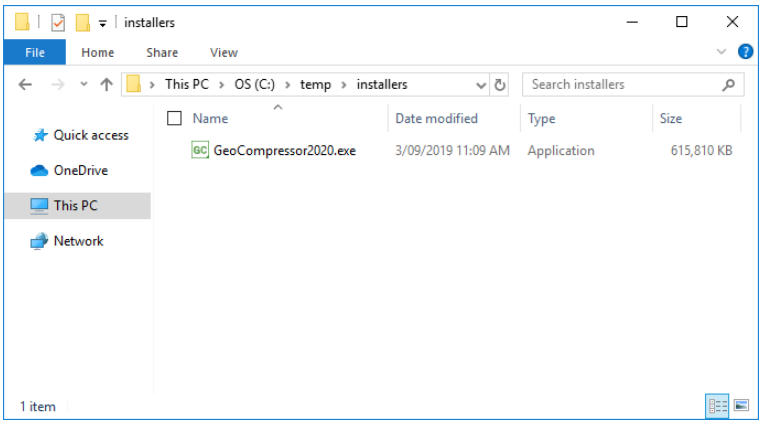
1. Other Linux distributions such as Debian, Ubuntu, Mint, Fedora and OpenSuSE are considered *viable* platforms and may require additional installation requirements.

Installation Guide

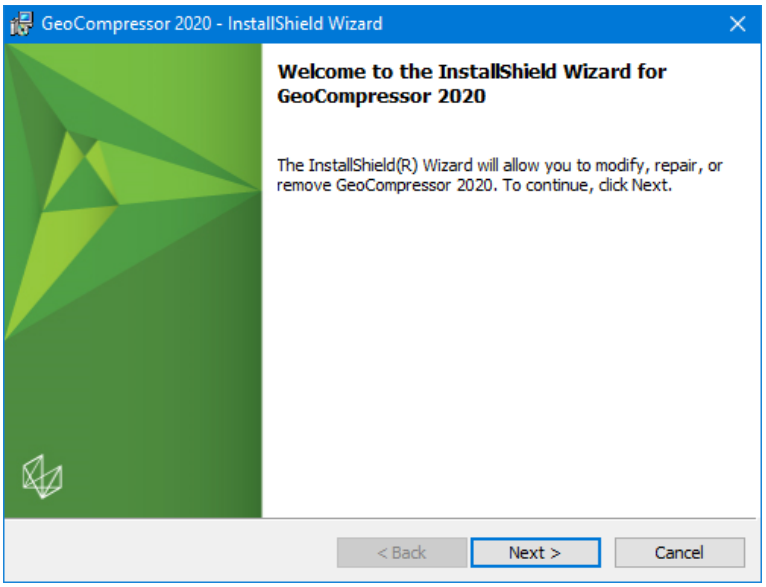
Windows

Installation

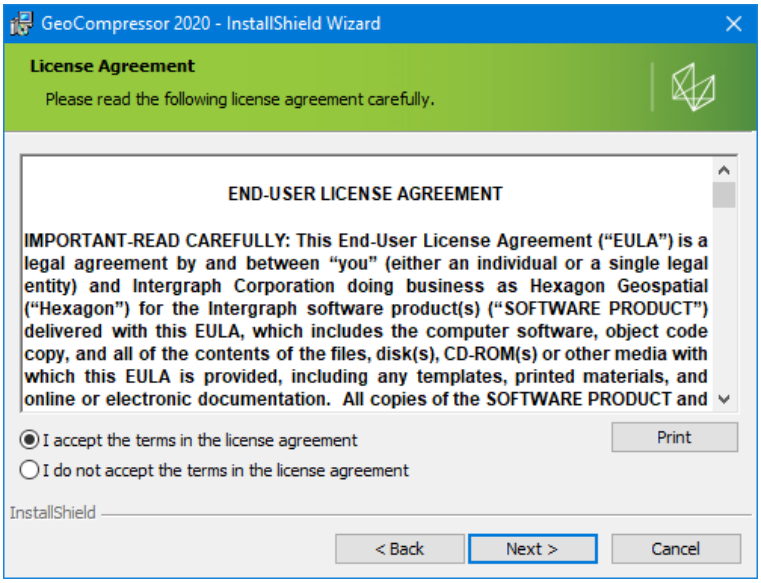
1. Download and execute the installation package from the product website.



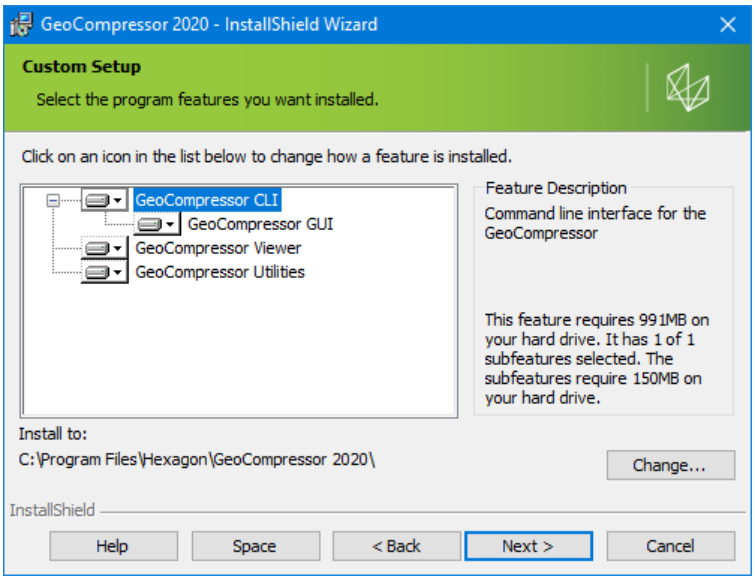
2. The installation wizard appears, click **Next** to continue.



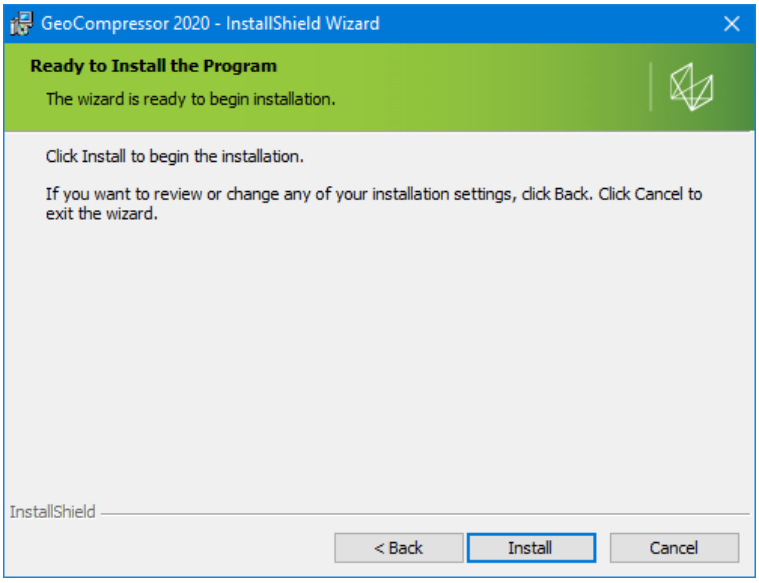
3. You may select **Complete** to continue, or **Custom** to customize the installation. Click **Next** to proceed.



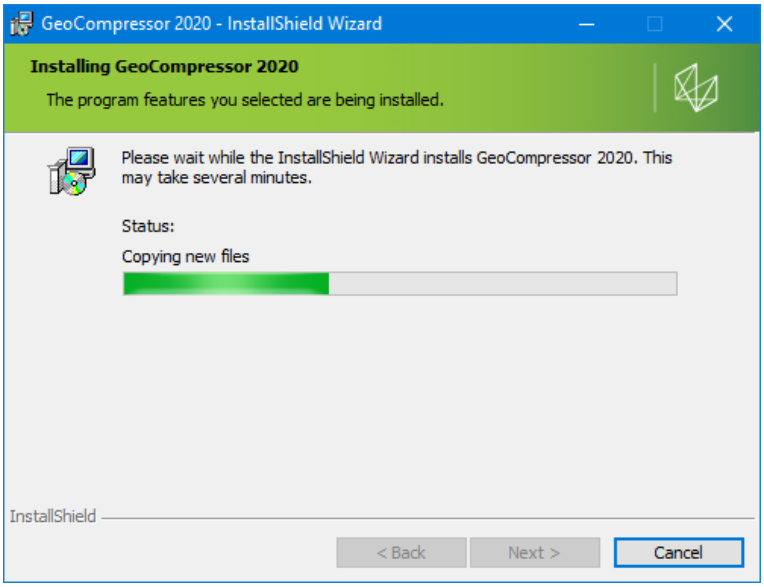
4. If you have selected **Custom** previously, then a list of features that can be added/removed is shown and has a **Change** button to specify a different installation directory.



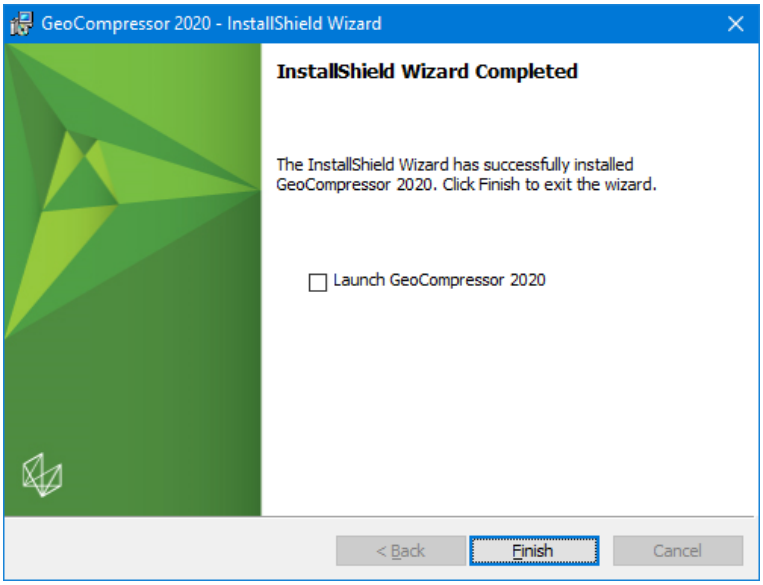
5. Click **Next** to continue.



6. Wait for the installation progress.



7. After progress, it informs the installation has completed. Click **Finish**.



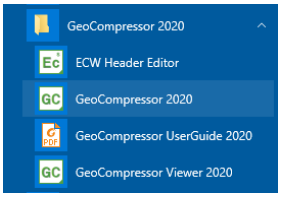
GeoCompressor can be executed via its three command lines:

"C:\Program Files\Hexagon\GeoCompressor[year]\bin\ImageCompressor.exe"

"C:\Program Files\Hexagon\GeoCompressor[year]\bin\PointCompressor.exe" or

"C:\Program Files\Hexagon\GeoCompressor[year]\bin\MAppUploader.exe"

or via the Start Menu shortcut to launch the wizard user interface.



See Licensing and Usage Chapters for next steps.

Uninstallation

To remove or uninstall, use the standard Windows "Add/Remove Programs" process.

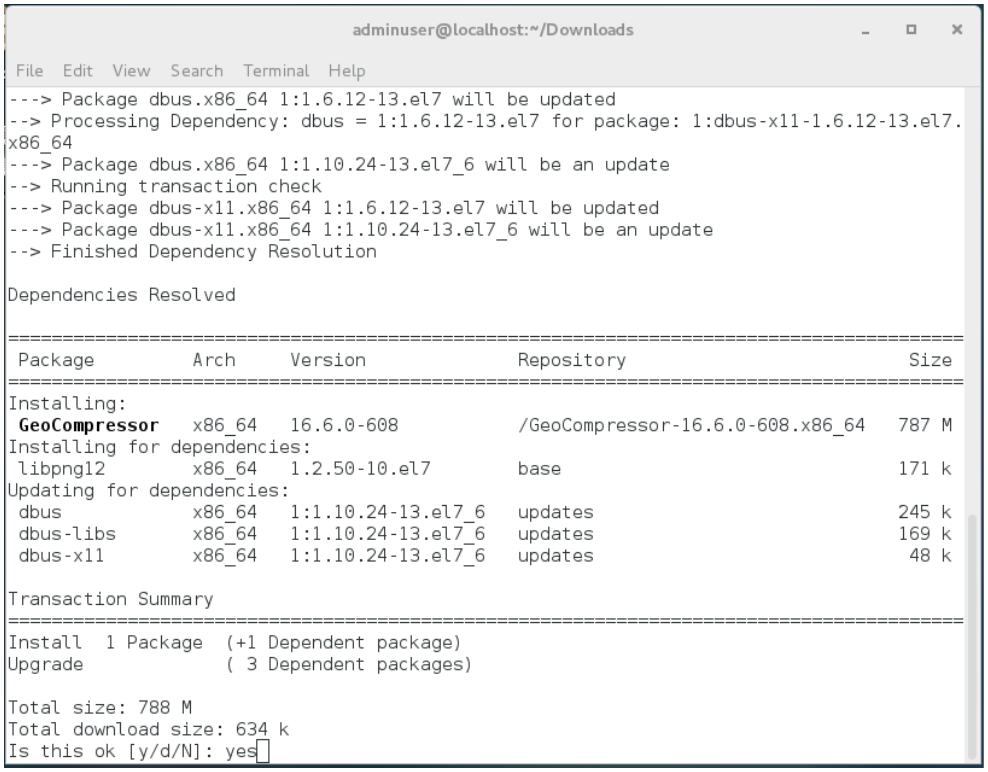
Linux

Installation

1. Download the installer from the product website and execute the .bin package
2. Review the EULA terms and type “yes” to continue and an RPM package will be extracted.
3. Execute the RPM.

   A. CentOS/RedHat/Fedora - using *yum install GeoCompressor.[version].x86_64.rpm*.

yum install will automatically determine dependencies before installing GeoCompressor in the system. Type ‘yes’ and enter to download any identified dependencies.



B. Ubuntu 16/18 – install dependencies and then use *alien*

1. *sudo apt-get install libpng12-0*
   a. if failed, append *'deb http://mirrors.kernel.org/ubuntu/ xenial main*' in the /etc/apt/sources.list. After adding the line, do *sudo add-apt-repository universe && sudo apt update*.
2. *sudo apt-get install libcurl4*
3. *sudo apt-get install alien* (skip this if installed)

4. *sudo alien -i --scripts GeoCompressor.[version].x86_64.rpm*

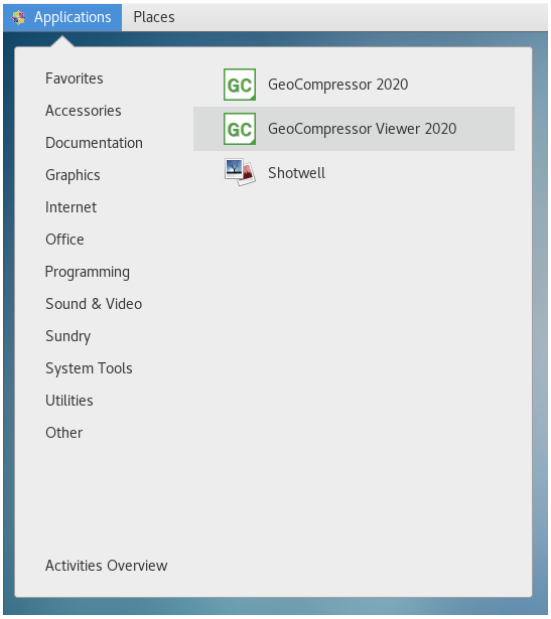
```
adminuser@adminuser-VirtualBox:~/Downloads$ sudo apt-get install libpng12-0
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  libpng12-0
0 to upgrade, 1 to newly install, 0 to remove and 3 not to upgrade.
Need to get 116 kB of archives.
After this operation, 285 kB of additional disk space will be used.
Get:1 http://mirrors.edge.kernel.org/ubuntu xenial/main amd64 libpng12-0 amd64 1.2.54-1ubuntu1 [116 kB]
Fetched 116 kB in 4s (27.4 kB/s)
Selecting previously unselected package libpng12-0:amd64.
(Reading database ... 166291 files and directories currently installed.)
Preparing to unpack .../libpng12-0_1.2.54-1ubuntu1_amd64.deb ...
Unpacking libpng12-0:amd64 (1.2.54-1ubuntu1) ...
Setting up libpng12-0:amd64 (1.2.54-1ubuntu1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
adminuser@adminuser-VirtualBox:~/Downloads$ sudo alien -i --scripts GeoCompressor-16.6.0-611.x86_64.rpm

warning: GeoCompressor-16.6.0-611.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID 8ded2da7: NOKEY
warning: GeoCompressor-16.6.0-611.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID 8ded2da7: NOKEY
warning: GeoCompressor-16.6.0-611.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID 8ded2da7: NOKEY
```

4. Default installation location is *"/usr/local/hexagon/geocompressor[year]/"*. Ensure the user installing the application has appropriate rights to write to this location.
5. The installation has completed.

GeoCompressor GUI can be executed via command line from:

"/usr/local/hexagon/geocompressor[year]/bin/GeoCompressor.sh"

or via the Application shortcut in the "Graphics" category on KDE and Gnome Window managers.



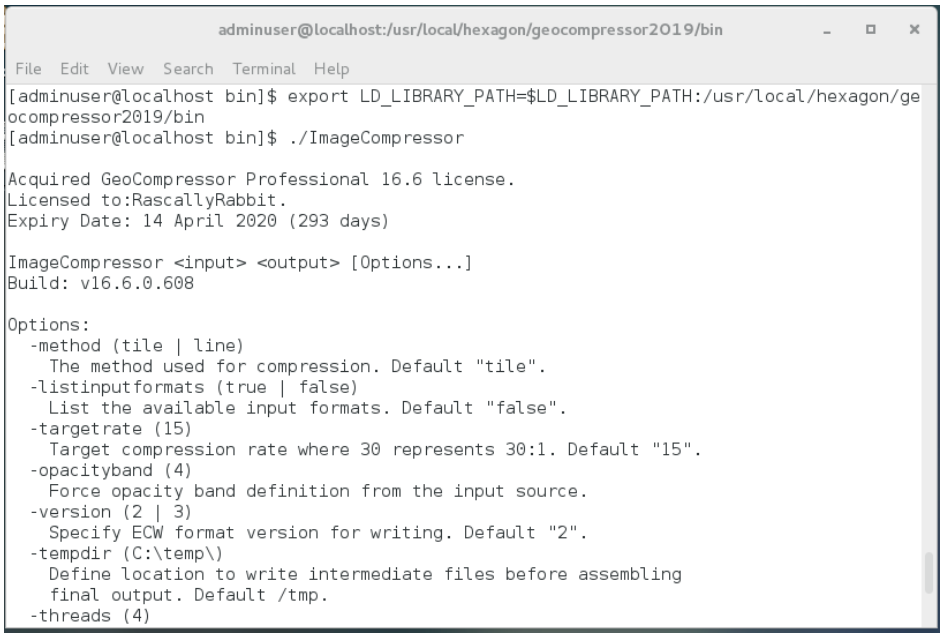
GeoCompressor CLI components can be executed via command line from:

"/usr/local/hexagon/geocompressor[year]/bin/ImageCompressor"

"/usr/local/hexagon/geocompressor[year]/bin/PointCompressor"

"/usr/local/hexagon/geocompressor[year]/bin/MAppUploader"

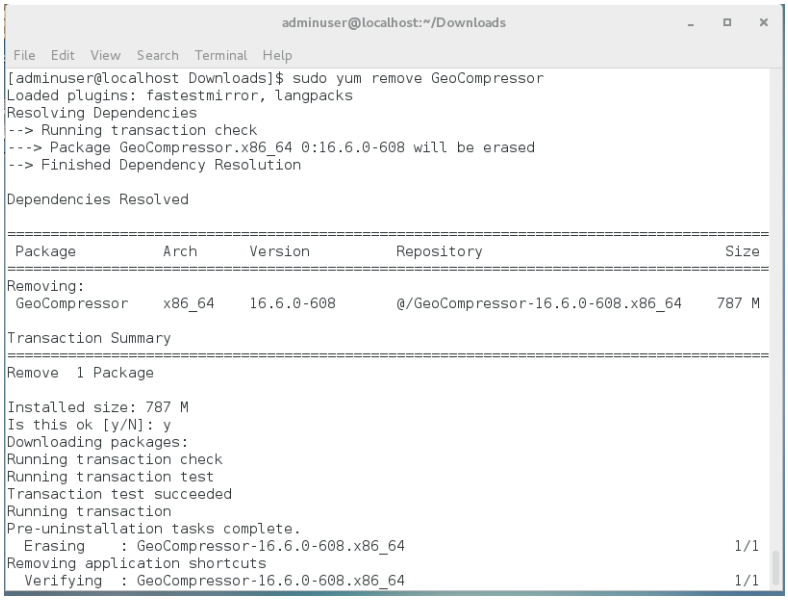
To run these CLI components, its recommend to use the paired shell scripts such as ./ImageCompressor.sh to set the required environmental paths.



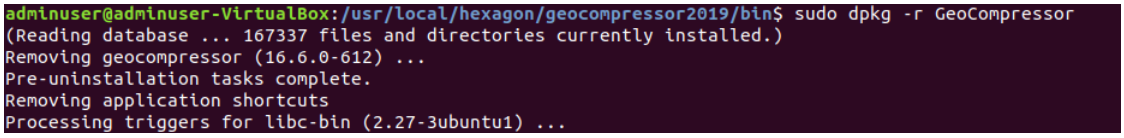
See Licensing and Usage Chapters for next steps.

Uninstallation

A. CentOS/RedHat/Fedora use *yum remove GeoCompressor*.



B. Ubuntu use *dpkg -r GeoCompressor* command.



Licensing

Overview

GeoCompressor is designed to complement existing ERDAS APOLLO installations and provide a decoupled, rapid compression tool. The product is Dual-licensed with respect to the product codes to meet both use-cases.

Product Family	Required License Level	Version
GeoCompressor	Essentials	16.6
GeoCompressor	Advantage	16.6
GeoCompressor	Professional	16.6
ERDAS APOLLO	Essentials	16.6
ERDAS APOLLO	Advantage	16.6
ERDAS APOLLO	Professional	16.6

If node-locked or concurrent licenses are unavailable for any of the above products and versions the compressor will fail. Where certain functionality is only available at a particular licensing tier and an operation is attempted, a license error will be returned. Refer to functionality matrix below for details.

In the event only an ERDAS APOLLO license is found, GeoCompressor will unlock the same functionality found at the GeoCompressor Professional level. For example, ERDAS APOLLO Essentials will unlock unlimited compression found at GeoCompressor Professional. This relationship however is not bi-directional, GeoCompressor licenses will not unlock ERDAS APOLLO.

Host based licensing in the product ensures that if any of the above feature codes are already acquired on the same machine GeoCompressor will validate but *not* check out an additional license. For example, GeoCompressor and ERDAS APOLLO Advantage can coexist under one ERDAS APOLLO Advantage license when run on the same machine at the same time. Equally, you can run parallel GeoCompressor processes on the same machine while using only one concurrent GeoCompressor license.

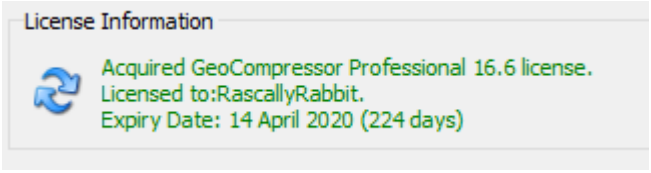
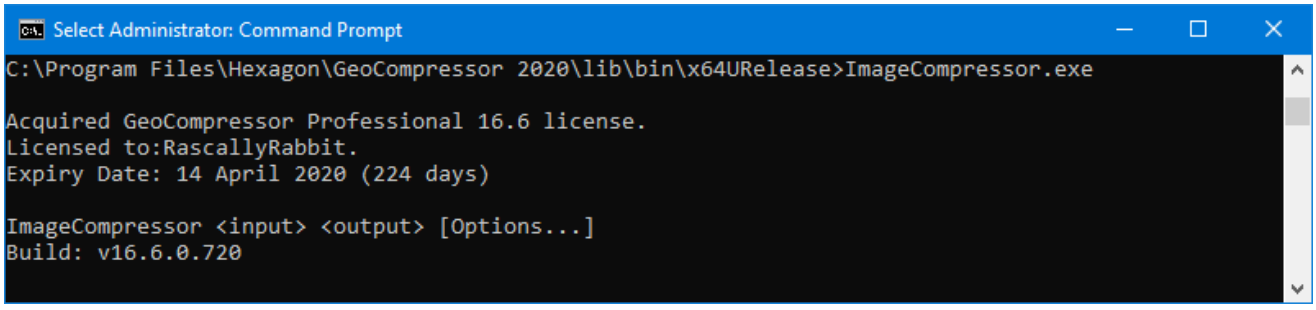


Figure 1 – License status in UI and Command line



Functionality Matrix

The features available at each GeoCompressor license tier are as follows.

	GeoCompressor		
	Essentials	Advantage	Professional
ECW Compression thread limits	4	8	Unlimited
Image Compression < 250 gigapixels	✔	✔	✔
Image Compression < 500 gigapixels	✘	✔	✔
Image Compression Unlimited	✘	✘	✔
Image Mosaicking [Up to gigapixel limit]	✔	✔	✔
Batch Image Compression [Up to gigapixel limit]	✔	✔	✔
Point Compression Unlimited	✘	✘	✔
ECW v3 Region Update	✘	✘	✔
Concurrent License only	✔	✔	✔
Subscription-only	✔	✔	✔

⚠ The size of a file in gigapixels can be calculated by multiplying the number of rows (height) by the number of columns (width). For example, an image with 20,000 rows and 20,000 columns would equal 400,000,000 pixels or 0.4 gigapixels. The number of bands, cell type or the amount of “empty” areas are not considered.

Installation and Configuration

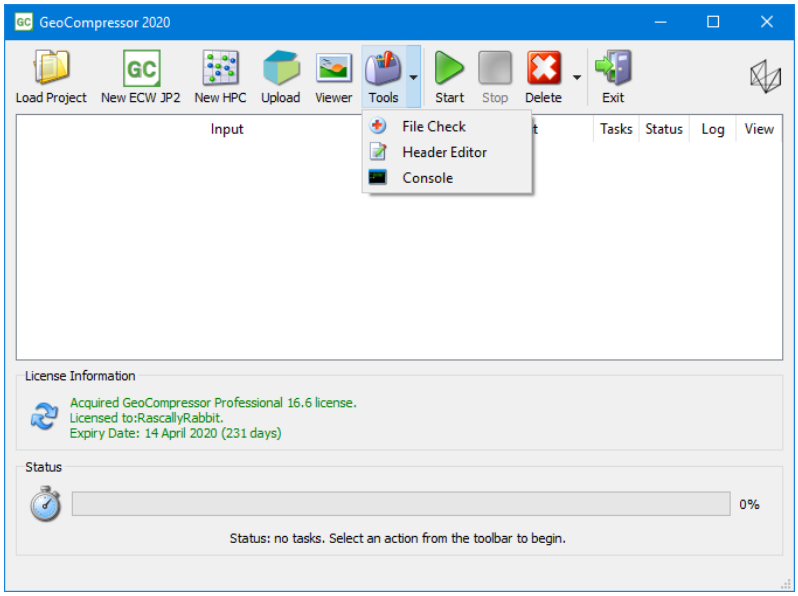
Refer to Hexagon Geospatial Licensing Portal for installation instructions and troubleshooting information for the Licensing server technology. Windows and Linux licensing information is available.

For most users with an existing product, no additional licensing steps should be required as the dependant product license codes should already be configured.

GeoCompressor Usage

Overview

GeoCompressor task window will list all previous and current compression tasks that have been defined and is the main user interface control outside of the direct command line interface.



The toolbar icons on the main interface are as follows:

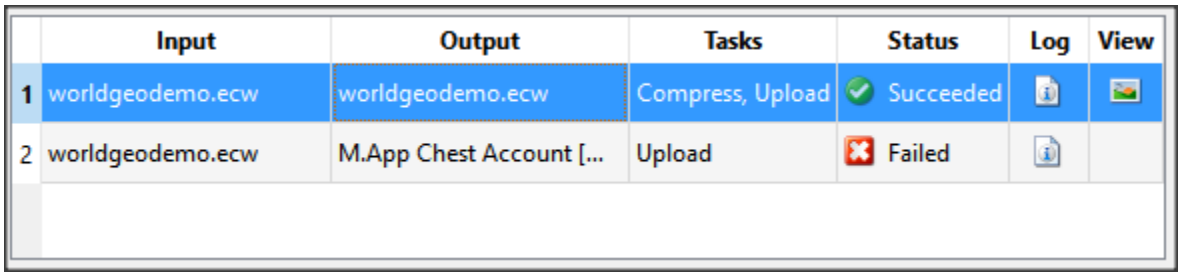
- “Load Project” – Load a previously defined compressor XML project.
- “New ECW JP2” – Run the Image Compressor wizard to compress files.
- “New HPC” – Run the Point Compressor wizard to compress a point cloud file.
- “Upload” – Upload a data file to a Smart M.App Cloud storage account.
- “Viewer” – Run the GeoCompressor Viewer application to display ECW or JPEG 2000 images.
- “Tools” – A tool menu of utilities comprising of the following:
  - “File Check” - check the integrity of an ECW or JPEG 2000 file.
  - “Header Editor” – edit the georeferencing or metadata information stored in an ECW or JP2 file.
  - “Console” – Open a terminal/console and run command line utilities such as NCSFileInfo.exe or the ImageCompressor.exe directly (Windows only).

Refer to the following chapters below for more information.

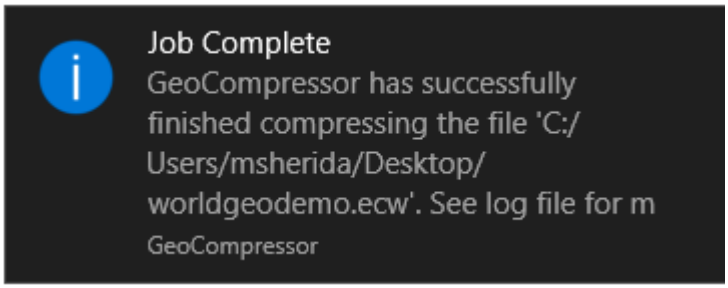
- “Load XML” allows predefined GeoCompressor XML files that describe an image compression task to be loaded and run directly, bypassing the wizards.
- “Delete Task” removes the selected tasks from the main window, for example to clear all previous jobs.
- “Start” and “Stop” buttons allow you begin or cancel currently running jobs but note there is no resume. Once a task is stopped all output and temporary files are removed and it must be restarted.

- “Exit” closes GeoCompressor fully and does not minimize to the system tray (default behavior).
- “About” confirms build, licensing, EULA and acknowledgement information.
- GeoCompressor Viewer.

In the event of a failed task, the status will be reported and a link to the log-file for easy review. On a successful compression task, the “View” column also becomes active and allows you to open the output file with the registered system application to open ECW, JPEG2000 or HPC files, or open a web browser to your M.App Chest page for successful uploads.



GeoCompressor has a system tray icon and will report there via a pop-up balloon the status of the completed task (success or failure).



Additionally, if you right click the tray icon, there are further options to hide/show the compressor, or launch a wizard for a new task.

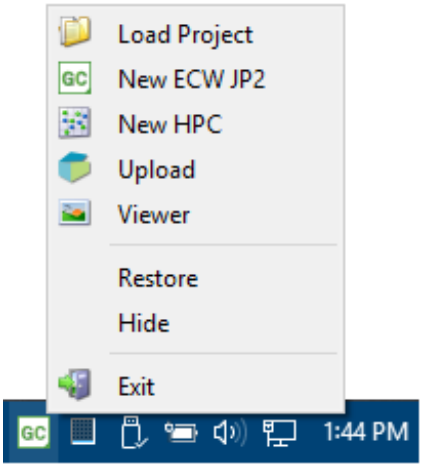
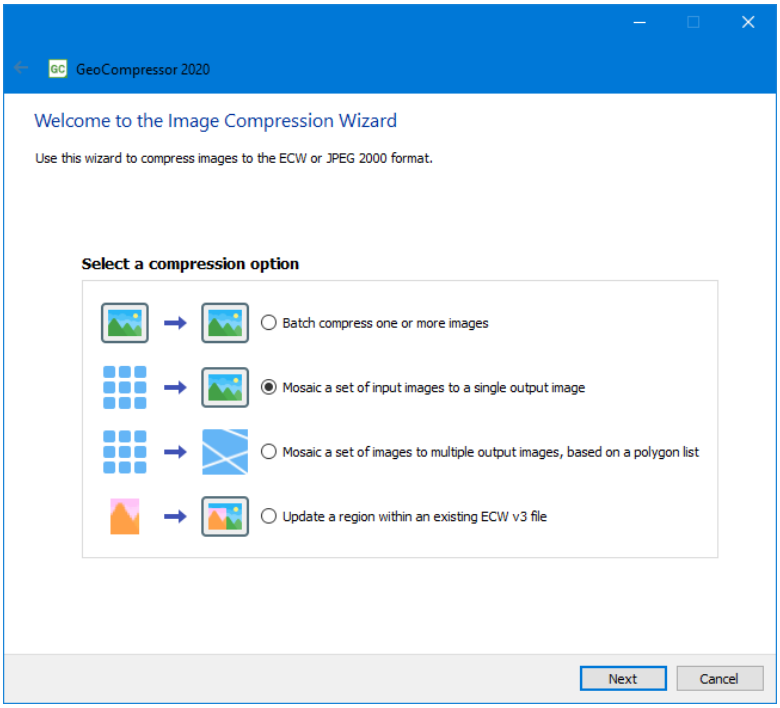


Image Compressor Usage

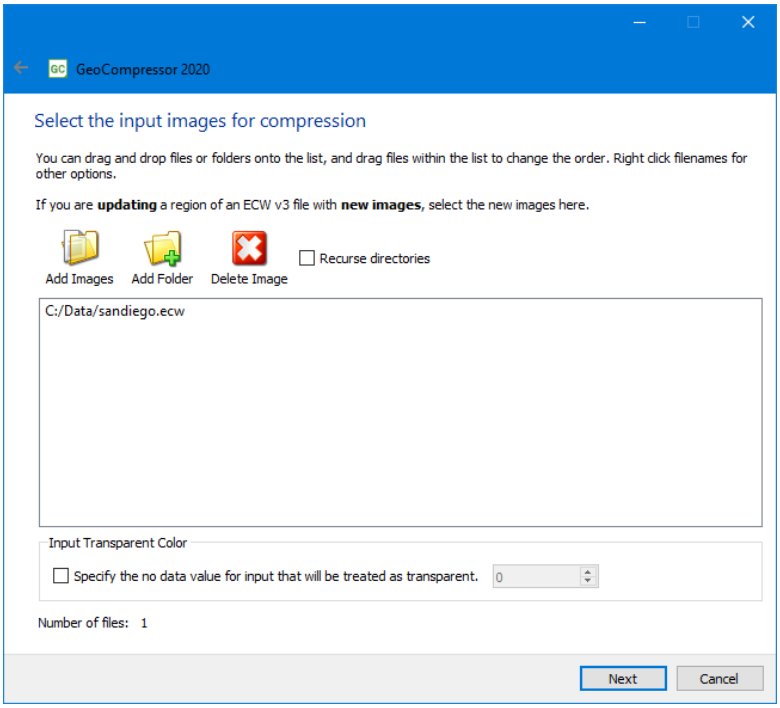
Wizard Mode

1. ECWJP2 Task wizard is comprised of 4 separate task types.



   a. For “Batch compress one or more images” GeoCompressor will create an output image for each input image. This provides a rapid way to compress thousands of images in batch.
   b. For “Mosaic a set of images to a single output image”, GeoCompressor will combine all selected images into a single compressed output image. This mode was designed for tiled, preferably adjoining images. The mosaic is created on the fly as part of the compression process, no intermediate files are created.
   c. For “Mosaic a set of images to multiple output images, based on a polygon list”, the compressor will create a mosaic (as in option b) but will output multiple images, based on a shapefile that contains multiple polygons that define the output extents of each image.
   d. Use “Update a region within an existing ECW v3 file” to selectively update only a portion of an existing ECW file. This is particularly powerful for terapixel sized ECW’s to refresh only a portion of the image with new imagery, without recompressing the whole file.

2. Regardless of the type of task selected, the next step is to select the input images.

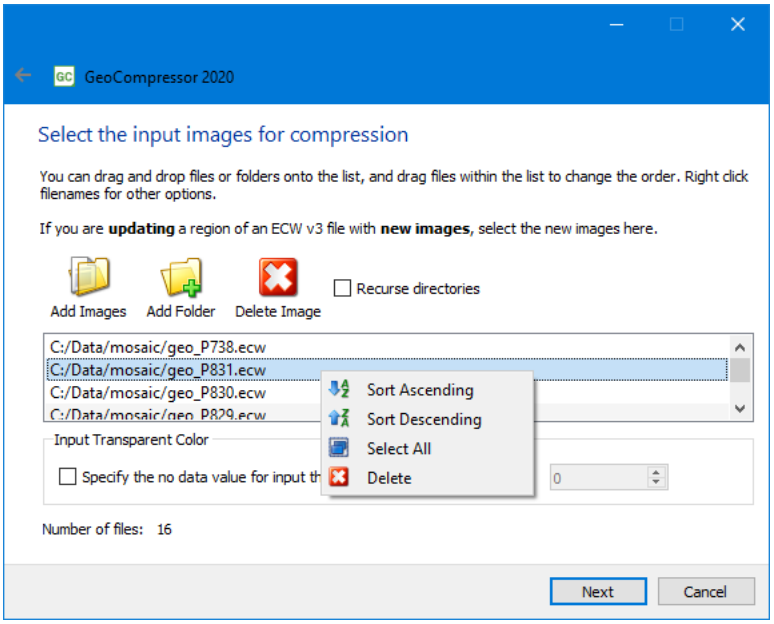


a. Add images allows individual file selection using the default “image filter” of common input file types (*.ecw, *.jp2, *.tif, *.alg, *.ers, *.img, *.vrt) or “All files” (Refer to supported Image Formats Chapter for full list).

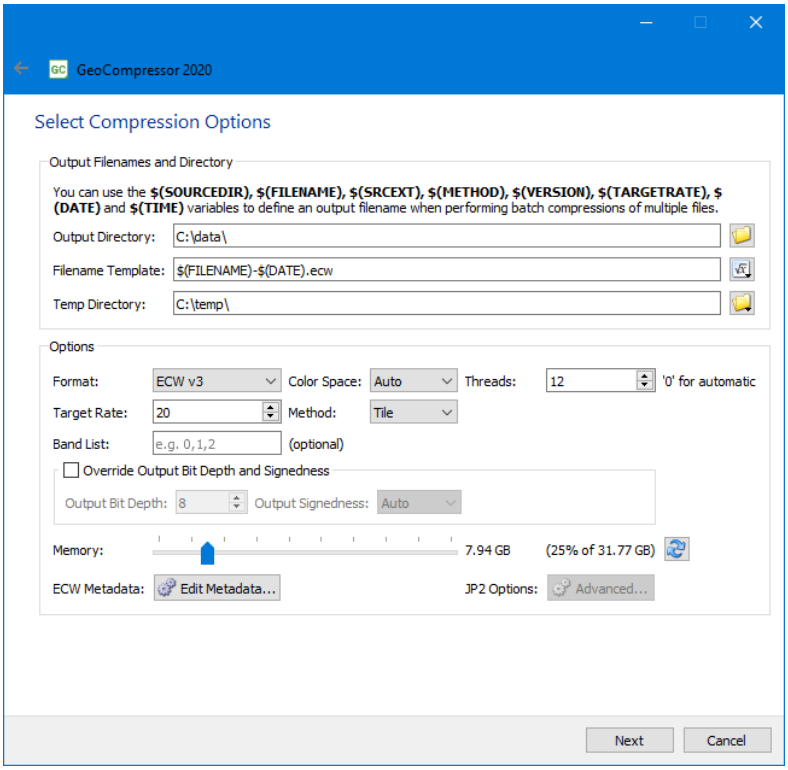
b. Add folder can be used in conjunction with the “Recurse directories” option to add thousands of files within a directory.

c. If the input files contain a transparent color (that is not recognised as a null data value by the compressor) then you may set the value here. The value will apply to all input bands (e.g. 0 would be black, 255 would be white for RGB images).

d. If mosaic wizard was selected, the order in which the datasets are displayed in this list box represents the display order of the output mosaic where datasets displayed at the “top” will be rendered on “top” of the output image. You may use the contextual right click menu or drag and drop to reorder the display order.



3. Standard compression options are now presented.



a. The output filename and directory can be defined using standard file locations or with templates, which are powerful ways to ensure the output names are unique.
b. The options listed are identical to the command line equivalent. Refer to the Command Line Options chapter below for more information.
    i. Format: ECW or JPEG 2000
    ii. Color Space: RGB, Greyscale, Multiband or Auto (automatically selected from the source)

iii. Threads: the number of CPU threads to use during compression (tile mode only). ‘0’ will choose an appropriate number of threads for the machine (2 x physical CPUs).

iv. Target Rate: the target compression ratio for the output file. This higher the ratio, the more compression, but the more visually lossy the image becomes.

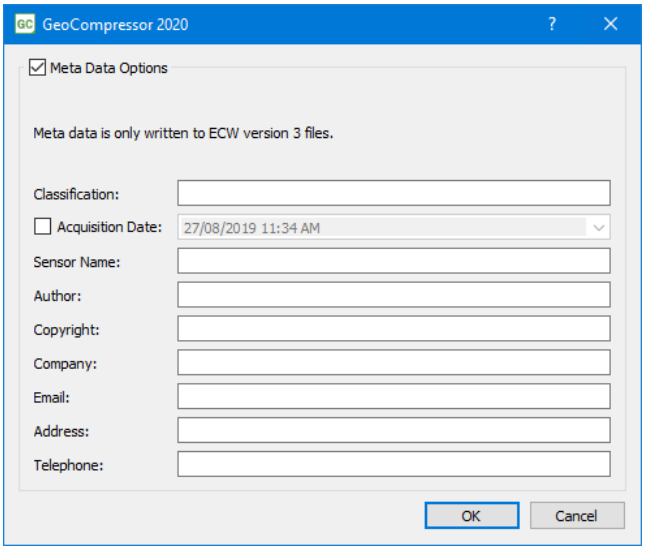
v. Method: Tile or Line. Tile based compression is faster and uses multiple threads, this is the default. The Line based compressor is single threaded but may be required for particular data types.

vi. Band List: Hard code the number of bands in cases where the compressor cannot work out the correct RGB bands from the input data. The index is zero based.

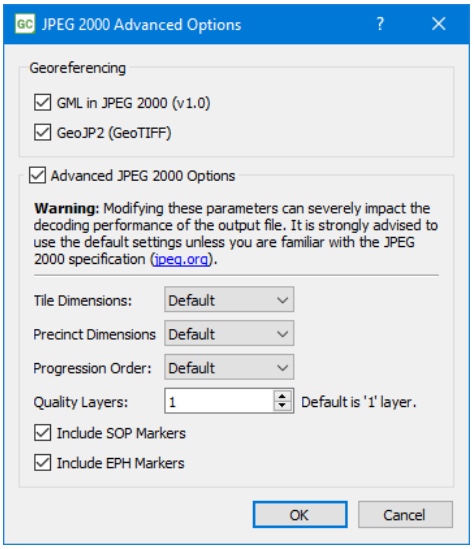
vii. Override Output Bit Depth and Signedness: Specify a different output bit depth and sign from the input dataset(s). This may be required where the compressor is unable to determine the correct output range, or the input data range is smaller than the full output type (e.g. 12-bit data stored in UInt16 data type), in which case overriding this may help to achieve more optimal compression rates. The compressor will not scale up or down the data (e.g. UInt16 to UInt8) but will truncate. This is an advanced option and should be used with caution.

c. You can select the maximum amount of memory the compression process will attempt to use here. The default is 25% of physical RAM. For large compressors, larger values will be required.

d. Select the “Edit Metadata Defaults” when encoding ECW version3 files if you would like to add extra metadata regarding the source imagery to the files. The metadata window is shown below.



e. Select the “JP2 Options: Advanced…” button to customize the JPEG 2000 format specific options. You should only modify these parameters if you understand the impact they will have. Changing these parameters can impact the decoding performance of the output file, and in some cases, may produce a file that is not a valid JPEG 2000 image.



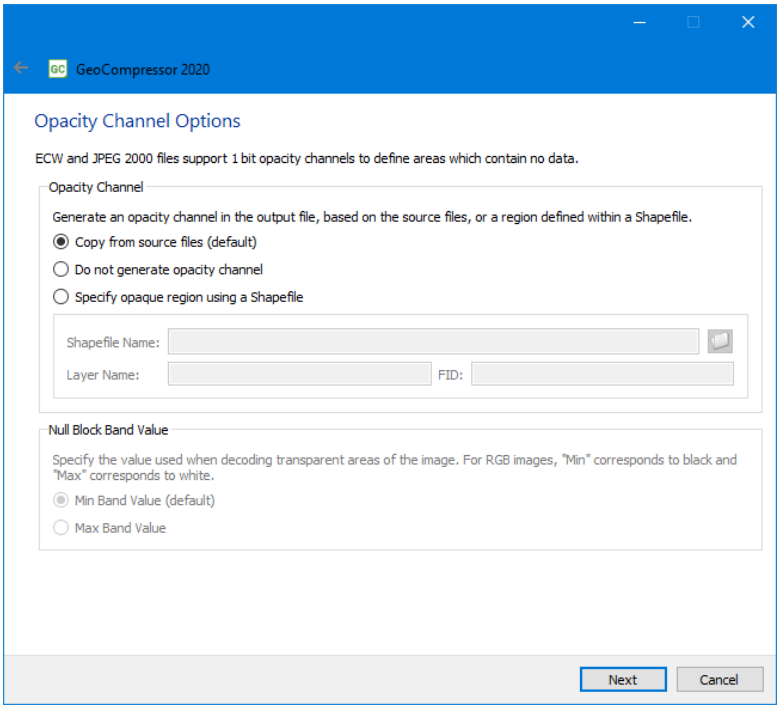
f. GML in JP2: Write an output box to the JP2 file containing the ISO standard GML in JP2 coordinate reference system reference box.

g. GeoJP2: Write an output box containing the common GeoJP2 standard (using TIFF tags) coordinate system reference box.

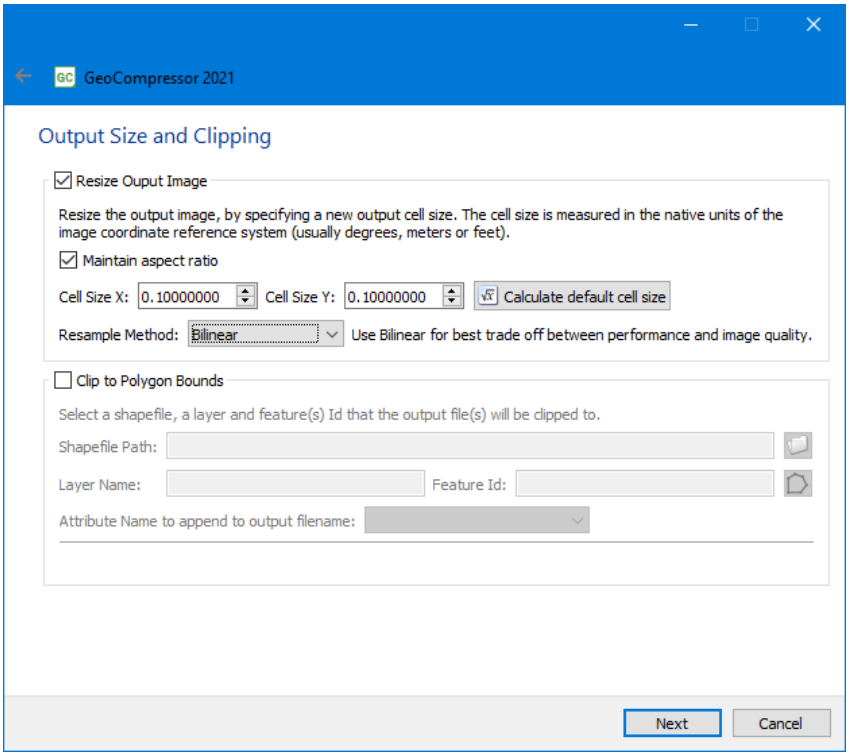
h. Advanced JPEG 2000 Options: adjust the layout parameters for the output JP2 file. Consult the JP2 documentation at https://jpeg.org/jpeg2000/ for information on the specific meanings of these parameters.

4. If using the ECW Update Wizard, the output options will be disabled as you cannot modify the compression parameters in this mode. The “Output File” represents the existing ECW v3 file as it will be updated in-place. No copies will be created so it is recommended to backup images when testing.

5. Opacity and null block options is the next configuration step. (Not available in Update mode)



6. Select Next to display the “Output Size and Clipping” page.



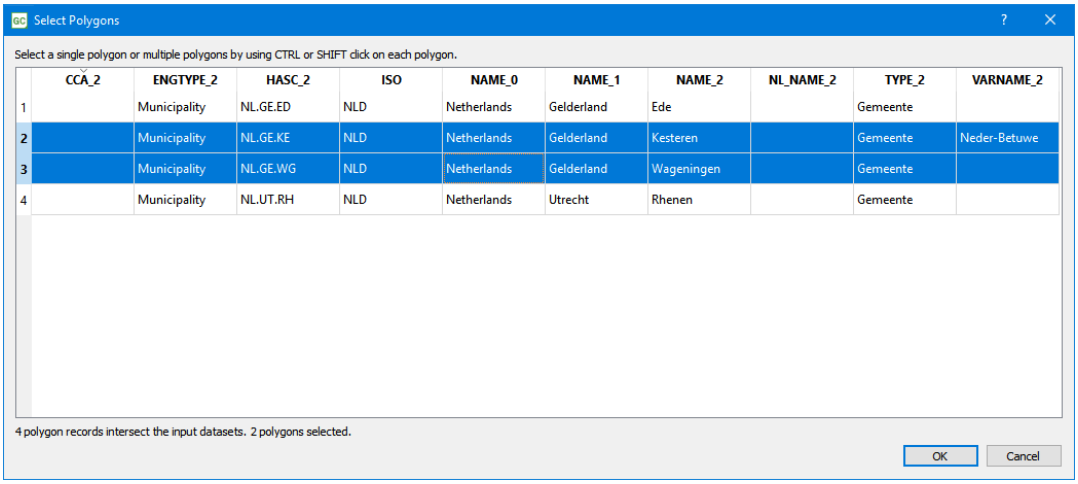
   a. Resize (resample) the output image by specifying a new cell size (in the natural units of the projection). The default input cell size is pre-calculated. Select the resample method to use when reading the data. The value can be one of Nearest Neighbour, Bilinear, Cubic, Cubic Spline, Lanczos, Average, Mode or Gauss. Nearest Neighbour is the lowest quality, but the fastest speed. Bilinear is recommended for most application types with medium speed and high quality. You should experiment with the method based on the data being compressed.

   b. Select “Maintain aspect ratio” to ensure the cell size remains uniform. Some data types may have non-square pixels, in which case unselect the checkbox and enter in the desired output values.

      Note: *not all viewing applications support images with non-square cell sizes.*

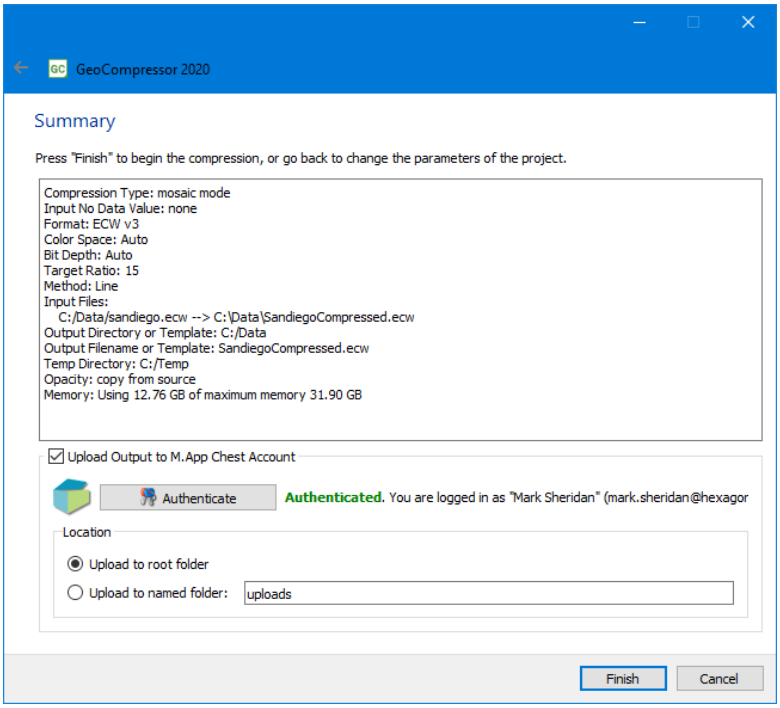
   c. Select a shapefile containing polygon to which you would like an output file per polygon. **Note**: *the shapefile must be in the same projection as the input image(s).*

7. Use the polygon selector to select only the polygons for whose extents you would like to generate an output file. The polygon will also be used for generating the opacity channel for the image.

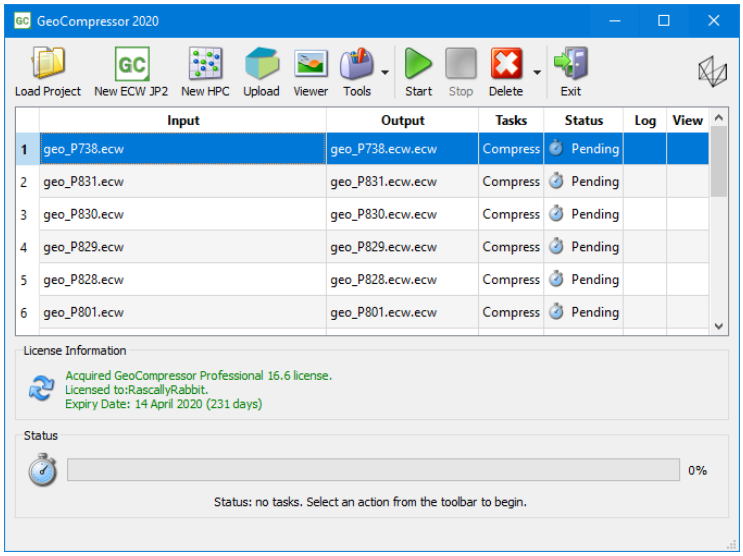


a. You may select one or more polygons. Take note of the table column header names. One of the attributes must be used when naming the output file, be sure to choose something unique. In this example above, use the “NAME_2” attribute in the output filename for easy identification of the images.

8. A summary page is shown to confirm the selected settings.



9. As part of finalizing the compression process, you may opt to upload the output to your Smart M.App Chest account. Select the “Upload output to your M.App Chest Account” to enable the feature. Refer to section “Upload Usage” on how to login and enter your credentials.
10. Select **Finish** to close the wizard and view the job in the task panel.
11. Click **Start** button to begin the compression jobs, which will then execute the jobs in sequential order. Executing jobs in parallel is not supported by the UI but can be achieved through direct command line access.


GeoCompressor 2020
Load Project
New ECW JP2
New HPC
Upload
Viewer
Tools
Start
Stop
Delete
Exit
Input
Output
Tasks
Status
Log
View
1 geo_P738.ecw geo_P738.ecw.ecw Compress Pending
2 geo_P831.ecw geo_P831.ecw.ecw Compress Pending
3 geo_P830.ecw geo_P830.ecw.ecw Compress Pending
4 geo_P829.ecw geo_P829.ecw.ecw Compress Pending
5 geo_P828.ecw geo_P828.ecw.ecw Compress Pending
6 geo_P801.ecw geo_P801.ecw.ecw Compress Pending
License Information
Acquired GeoCompressor Professional 16.6 license.
Licensed to:RascallyRabbit.
Expiry Date: 14 April 2020 (231 days)
Status
0%
Status: no tasks. Select an action from the toolbar to begin.

Command-Line Mode

The command-line mode offers fine-grain control over the compression process. Where possible defaults are set to ensure that in many cases only the <input> and <output> file is required however careful consideration of all options is recommended. When the user interface is used the output compression log includes the command line options provided. To see the command line options, open a shell or command prompt in the “bin” directory of the software installation and type “imagecompressor.exe” without arguments. The command line options are shown below.

Note: The available options are identical for both Windows and Linux versions of GeoCompressor unless otherwise specified.

Note: Not all options are available in command-line mode. More flexibility is achieved by using a compressor XML input file to describe the compression process.

```
C:\Program Files\Hexagon\GeoCompressor 2020\lib\bin\x64URelease>ImageCompressor.exe

Acquired GeoCompressor Professional 16.6 license.
Licensed to:RascallyRabbit.
Expiry Date: 14 April 2020 (225 days)

ImageCompressor <input> <output> [Options...]
Build: v16.6.0.714

Options:
  -method (tile | line)
    The method used for compression. Default "tile".
  -listinputformats (true | false)
    List the available input formats. Default "false".
  -targetrate (15)
    Target compression rate where 30 represents 30:1. Default "15".
  -opacityband (4)
    Force opacity band definition from the input source.
  -version (2 | 3)
    Specify ECW format version for writing. Default "2".
  -tempdir (C:\temp\)
    Define location to write intermediate files before assembling
    final output. Default C:\Users\ctweedie\AppData\Local\Temp.
  -threads (4)
    Number of CPU threads used for compression. Tile based algorithm only.
    Line will always use 1 CPU thread. Default value is CPU core count.
  -memcache (0.25)
    Percentage of total RAM allocated for compression where 0.25 represents
    25% of the system memory.

Region Options:
  -opacity (0 | 1 | 2 | 3)
    Manage the opacity band and/or fill transparent areas according to input
    region. 0 and 2 will force the generation of an opacity band in the output.
     - 0 generate an opacity band based on the input and a region definition.
     - 1 set transparent areas to -nullvalue based on a region definition.
     - 2 (default) does both 0 and 1.
     - 3 Strip the opacity band from the output if there is one in the input.
  -datasetregionfile (region.txt)
    ASCII file containing dataset coordinates defining a closed non-null
    polygon; see the user guide for more info.
  -shapefile (vector.shp,layername,fid)
    Esri Shapefile that defines a non-null polygon FID representing the
    dataset active area.
  -wktfile (region.txt)
    OGC well known text that defines a closed non-null polygon.

Advanced Options:
  -bitdepth (8 | 16)
    Force output bitdepth to 8, 16. Default to input bitdepth.
  -signed (true | false)
    Force output to be signed. Default "false".
  -colorspace (greyscale | multiband | rgb)
    Force color space for the output. No value will retain input colorspace.
  -units (degrees | meters | degrees)
    Force units in ECW v2.  This value is only informational.
  -bandlist (0,1,2)
    Comma separated band indices to write to the output file. Default is null
    which retains all bands from input.
  -logfile (C:\temp\compression.log)
    Writes compression log file to the location.
  -loglevel (debug | info | warn | error | fatal)
    Log level setting. Default "info".
  -genstats (true | false)
    Generate statistics with full histogram as part of compression. Valid for
    ECW v3 output only. Default value "true".
  -inputnodata (0)
    Set up NoData value that will be applied to all bands.
  -nullvalue (min | max)
    min is 0, max is the largest value available for each band. Default "min".
  -srs (EPSG:2180)
    sets destination spatial reference system to the one specified. Note: this
    will not reproject the image.
  -xml (c:\data\mosaic.xml)
    Use the XML file as input for the compression process. The contents of the
    XML overrides all other command line options.
  -json (c:\data\mosaic.json)
    Use the JSON file as input for the compression process. The contents of the
    JSON overrides all other command line options.
  -generatexmltemplate
    Generate a generic compression configuration template in xml format.
  -generatejsontemplate
    Generate a generic compression configuration template in json format.
  -validatexml (c:\data\mosaic.xml)
    Validate xml file as input.
  -validatejson (c:\data\mosaic.json)
    Validate json file as input.
  -compressionheaps (number greater than zero)
    The number of heaps the compressor should use internally.
  -decompressionheaps (number greater than zero)
    The number of heaps the ecw/jp2 decoder should use internally.
  -openoptions (option1=value1,option2=value2)
    Passes options to the library opening the input file.
```

Input

For supported input format types see Supported Input Formats chapter. To compress any of these formats pass in the file location and the Image Compressor will attempt to load it via one of the supported Input Readers. If reading fails, the compressor will exit. The File location should be enclosed in quotes if the path contains spaces.

Output

Starting at version 2020 Update 2, the default format for ECW output is version 3, and the internal layout of ECW version 3 files now use a larger internal block size (128x128). This has some side effects, such as on average, the overall compression time is faster (up to twice as fast), however, the amount of memory required to complete the compression will be approximately 80% more (as the block size is doubled). The major advantage to this change, however, is a reduction of around 10-15% in storage space. The quality and decoded image are unchanged.

GeoCompressor supports writing ECW and JPEG2000 files where the type is defined by the output file extension. To write JPEG2000 the output file should be specified as C:\output\compressed.jp2 or for ECW C:\output\compressed.ecw. Based on the file type different compression options will become available according to the following format capabilities.

Capability	ECW v2	ECW v3	JPEG2000
Line compression	✔	✔	✔
Tile compression	✔	✔	✘
8-bit unsigned	✔	✔	✔
16-bit unsigned	✘	✔	✔
16-bit signed	✘	✘	✔
Visually lossless	✔	✔	✔
Numerically lossless	✘	✘	✔
Null block support	✘	✔	✘
Opacity band support	✔	✔	✔
Data statistics, histogram	✘	✔	✘
RPC storage	✘	✔	✘
Custom metadata	✘	✔	✔[3]
Region update	✘	✔	✘
Geo-referencing	GDT	GeoTIFF Tags	GML in JP2, GeoJP2
Colour space support	Greyscale, RGB, Multiband	Greyscale, RGB, Multiband	Greyscale, RGB, Multiband
Largest known image[4]	32 terapixels	48 terapixels	756 gigapixels

[3] Partial. Custom metadata can be written to JP2 UUID boxes however clients will have to detect its presence and parse the contents.

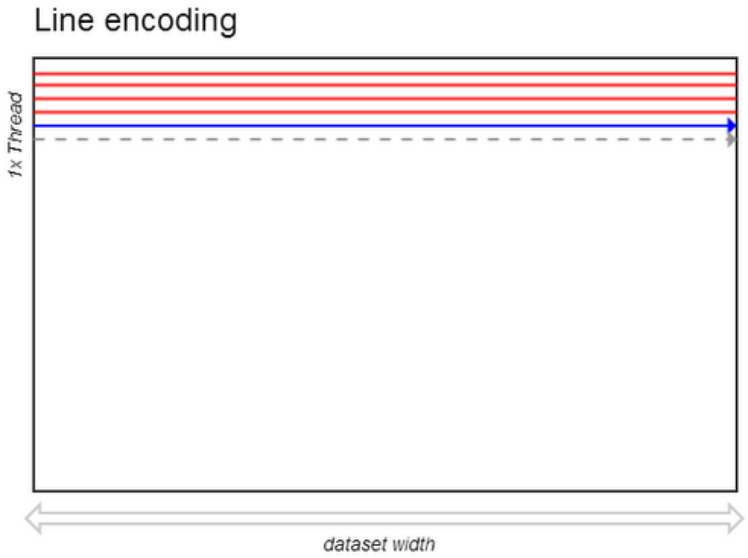
[4] Compressed using GeoCompressor, as at January 2014.

Options

-method line|tile

The ImageCompressor implements two types of image compression algorithms.

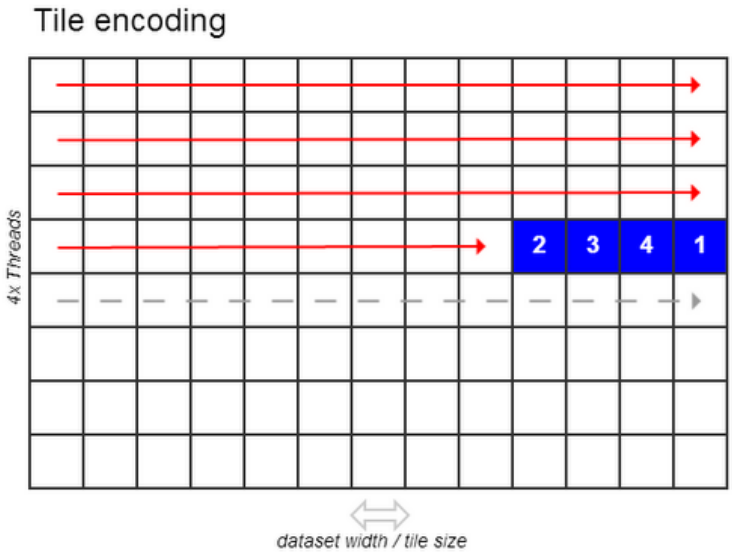
“Line” represents a scan-line based reader that reads across each scan-line, compresses and continues to the next scan-line until the end of file. This approach has been used since ECW’s inception.



Line is inherently single-threaded and cannot scale across multiple CPU cores however it benefits from lower memory requirements to compress. Line is particularly suited for scan-line structured input data such as Strip TIFF or file formats that do not perform well with multi-threaded readers.

Note: Line must be used for the compression of JPEG2000 files.

“Tile” is a new parallel algorithm introduced in the ERDAS ECWJP2 SDK v5 that reads input data in discrete tiles across multiple reader threads specified by the –threads option. Each thread processes independently in a thread-pool across the width of the dataset and is then repeated down the image.



The new algorithm results in scaling across CPU Cores however it requires more memory to compress the same input as “line”. There is a further trade-off with Disk I/O as the concurrent threads increase load and requires more data to be processed than line, increasing the likelihood of reaching a disk bottleneck. It’s also possible that the input Data Readers have not been optimized for multi-threaded reading creating additional bottlenecks. See FAQ for formats that are not thread safe.

Fast I/O is a requirement in order to feed data to the multiple worker threads otherwise CPU Utilization will be low, and performance may be slower than the line algorithm. Where Data I/O is sufficient, the tile algorithm can be more than 400% faster than line depending on hardware and input format used.

Both algorithms are suited for different situations. In order to determine the ideal method for your situation, benchmarks should be performed particularly where compression speed is an important measure.

Note: Output ECW files are binary identical irrespective of the compression method used.

-listinputformats true

Retrieves list of supported input formats across all data readers. See Supported Formats chapter for a formatted version. This option is mutually exclusive from all others.

-targetrate 15

The target compression rate is expressed as a ratio, where 15 represent “15:1” or in other words a 94% size reduction. The compressor uses this target to maintain image compression quality, which can result in the “Actual Compression Ratio” being higher or lower than the provided value. This is expected and is commonly seen on datasets with nodata areas or other areas with spectral characteristics that are highly compressible without sacrificing quality. A 15:1 target compressed image always produces comparable image quality to any other 15:1 target image even though actual rate may vary.

For RGB or Multiband input, 15:1 to 20:1 target compression rates produce visually lossless output results. For greyscale, 10:1 is recommended. Watch How to Save Time and Storage Space with Industry-Leading Imagery Compression for more information.

Note: To compress numerically lossless JPEG2000, specify a targetrate value of 1. Any other value will revert to the lossy compression type. ECW does not support numerically lossless compression so a targetrate of less than 5 is not recommended.

-opacityband 4

The compressor will by default attempt to retain any opacity or alpha band present in the input file. This option can be used to force the detection of the opacity channel if it cannot be determined automatically from the input reader. For example, where an input 4-band image is tagged as Bands # 0 to 3, where Band #3 is actually a mislabelled opacity band.

-version 2 | 3

The ECW file format has two versions available. ECW v2 is the legacy format with the widest industry support and ensures interoperability with all existing ECW software. ECW v3 was introduced in 2012 and adds new capabilities such as null blocks for improved performance and space savings, improved metadata storage and support for uint16 cell types.

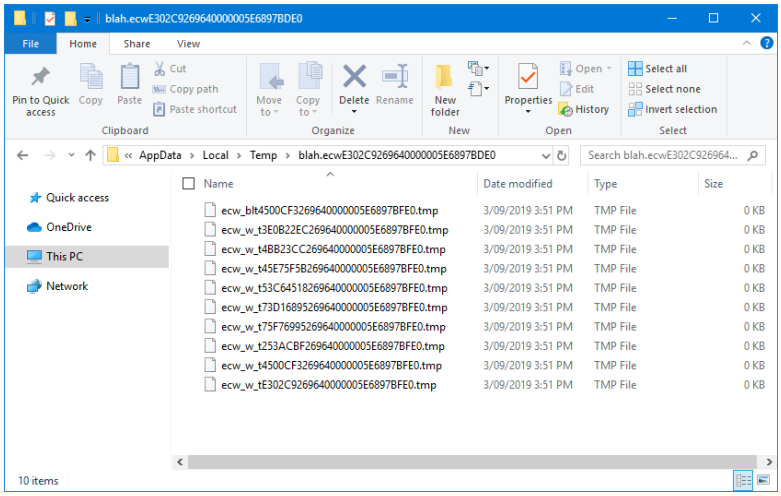
Defaults to ECW v2. This option is ignored for JPEG2000 output.

-tempdir C:\temp\

As part of every compression process the GeoCompressor must maintain a certain amount of intermediate data on disk before reassembling to form the final output format. These intermediate files are all stored in the tempdir location that defaults to the system %TEMP% environment location.

For performance reasons it's strongly recommended to write temporary data to a different physical disk than the input or output drives to maintain throughput. Users should allocate the same amount of storage at tempdir as at the final target destination. For example, if a 10 GB ECW output is estimated then at least 10 GB is required at the tempdir location.

Figure 2 – Example ECW temp files written to tempdir



GeoCompressor manages the files written to this location. On completion of all compression jobs the temp files will be deleted however if the compression process is interrupted temporary files may be left on disk. If no compression is currently running it is recommended to delete any orphaned temp files.

-threads 4

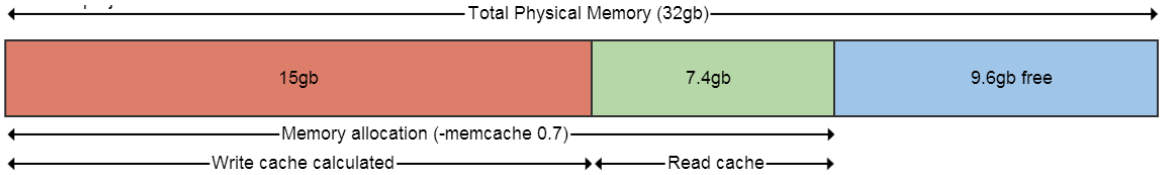
The number of threads used when compressing via the “tile” compression method. Defaults to the detected CPU Core count on 1 Processor but can be adjusted to determine ideal concurrency level. Like memcache using all threads may not yield best performance particularly if Disk I/O is inadequate. On large deployments with 32 cores or more, optimal throughput may be found setting threads to 8 or 16. Testing is always recommended.

Value is ignored when –method is set to “line” as it will always be one. See –method above for more information.

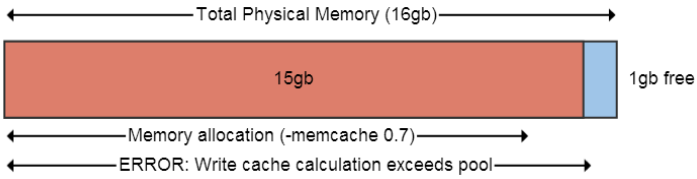
-memcache 0.25

Allocate a percentage of total System Memory to a memory pool for the compression process. GeoCompressor will calculate the required memory for the “Write Memory Cache” based on the input image dimensions and compression method specified (see table). Any remaining memory in the pool will then be assigned to the “Read Memory Cache”.

On a machine with 32 GB System Memory for example, a memcache of 0.7 can be visualized as follows:

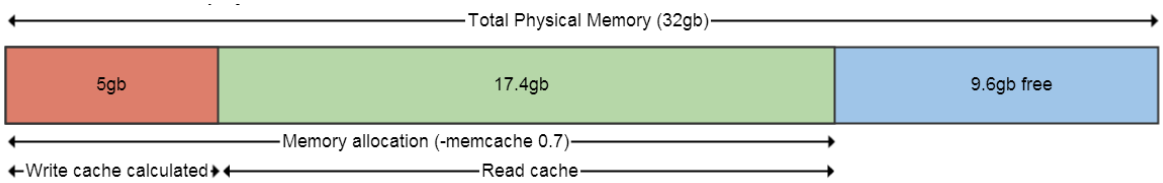


If the same input image was then compressed on a machine with only 16GB System Memory, the same memcache value of 0.7 would yield:

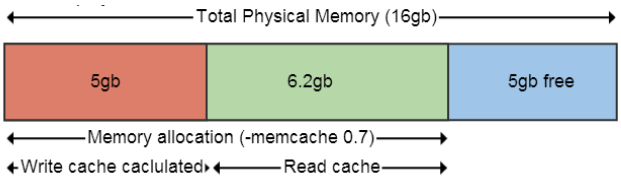


This would result in an error since the fixed “Write Memory Cache” requirement of 15gb is larger than the whole memory pool budget (16gb * 0.7 = 11.2 gb). To remedy, a memcache value of 0.95 (15.2 GB) is required, which would require careful management to ensure the system does not begin to swap to pagefile. Adding additional memory would be recommended in this situation or switching to the line compression method to reduce the memory requirement but potentially sacrificing throughput performance.

To compress a smaller hypothetical image on the 32 GB machine with the same memcache value of 0.7, the write cache is reduced and the read cache size increases dramatically to take the remainder of the memory pool.



The smaller image will now also be able to be run on the 16gb machine as the Write requirements no longer exceeds the memcache budget:



The size of the “Read Memory Cache” is secondary and increasing this value does not always yield an increase in performance. In other words, if a –memcache of 0.25 is sufficient to compress your input image it is not always advised to specify –memcache of 0.9. It is dependent on the type of input format, how well that dataset driver responds to a higher read memory cache and disk I/O performance. In some cases, performance could drop given additional overheads managing a large cache verse simply reading data from disk.

It is always recommended to leave a memory buffer to ensure long running compression jobs do not reach 100% Memory usage. GeoCompressor can fluctuate within a degree of tolerance of the memory budget before purging, so assigning 99% memory is never advised. This buffer size also depends on other applications that may be running on the machine, which is not considered by default.

The following table highlights the fixed write memory cache requirements for different input characteristics.

Input image size	Bands	Gigapixels	Output Format	Compression Method	Write Memory Cache (MB)
50,000 x 50,000 px	3	2.5	ECW	Tile	**247**
	3	2.5	ECW	Line	**136**
100,000 x 100,000 px	3	10	ECW	Tile	**443**
	3	10	ECW	Line	**223**
500,000 x 100,000 px	3	50	ECW	Tile	**1,960**
	3	50	ECW	Line	**918**
	8	50	ECW	Tile	**5,150**
100,000 x 500,000 px	3	50	ECW	Tile	**443**
	3	50	ECW	Line	**223**
	3	50	JPEG2000	Line	**192**
	8	50	ECW	Tile	**1,070**
500,000 x 500,000 px	3	250	ECW	Tile	**1,960**
	3	250	ECW	Line	**918**
	3	250	JPEG2000	Line	**763**
	8	250	ECW	Tile	**5,150**
	8	250	JPEG2000	Line	**1,910**
1,000,000 x 1,000,000 px	3	1000	ECW	Tile	**3,880**
	3	1000	ECW	Line	**1,750**
	8	1000	ECW	Tile	**10,250**

Note:

- the drop-in memory when using line over tile compression method
- the drop-in memory when the width of the input changes, despite identical gigapixel value
- the large increase in memory requirements moving from 3 to 8 bands

- the small memory drop using our default JPEG2000 profile over ECW in line mode
- although not shown, write memory requirements do not change depending on the –targetrate. A target of 10:1 will require the same memory as 50:1.

Region Options

-opacity 0 | 1 | 2 | 3

The opacity value defines the behaviour of the input region defined by –datasetregionfile, -shapefile and –wktfile.

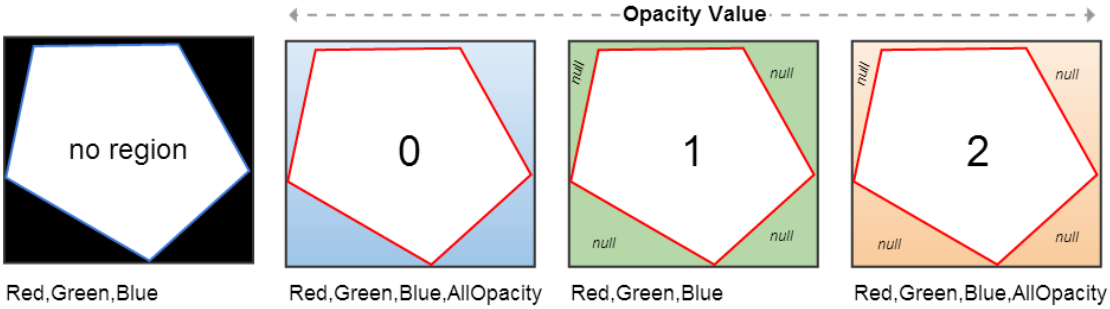
- -opacity 0: create an opacity band from the input
- -opacity 1: null blocks only *(ECW v3 only)*
- -opacity 2: opacity and null blocks *(if writing ECW v3)* [DEFAULT]
- -opacity 3: do not create an opacity band

When writing ECW v3, a value of 2 is recommended to benefit from an opacity band as well as reduced storage requirements and possible performance improvements using the null block capability. Be aware that null blocks introduce additional calculations in compression, so where the null area represents a small percentage of the total dataset area (Ratio to data), or the region complexity is large (a high number of vertices) a value of 0 may compress faster. Testing is recommended to quantify this difference for your situation. The output quality will be identical between 0 and 2. ECW users will be unaware whether null blocks are present and behave like any other ECW file.

When writing ECW v2 and JPEG2000 output, a value of 0 will ensure the output has an opacity band to make the background transparent.

If no region and no opacity value are specified, Image Compressor will recreate the input opacity information when detected however a region will override this automatic translation.

The following diagram visualizes each available option where the red polygon is the input region:



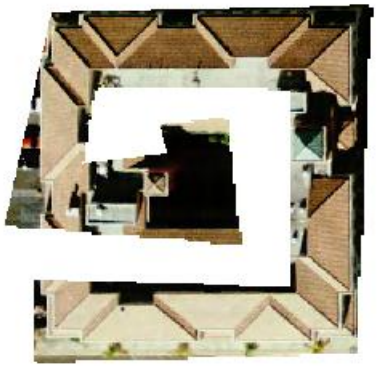
“No Region” or “Opacity 3” and “Opacity 1” have opaque backgrounds with no opacity band in the output. Opacity 1 will have reduced storage requirements due to null blocks however will still have a solid background colour (defined by –nullcolor). Only opacity values of 0 or 2 will create an opacity band and allow underlying data to be shown in this area.

Default value is “2” for ECW v3 output and “0” for ECW v2 and JPEG2000

-datasetregionfile region.txt

A file on disk that represents a closed polygon in image coordinates (pixel space) defining the active or data area of the image. All area outside of this polygon will be tagged as null and/or transparent as per the –opacity option.

GeoCompressor will not clip the output image dimensions to the extents of the input region so it should not be used to subset larger images.

Dataset coordinates	Output result
563,1642; ***1071,1681;*** ***1040,1408;*** ***899,1423;*** ***915,1486;*** ***727,1509;*** ***758,1337;*** ***1181,1345;*** ***1181,1775;*** ***625,1760;*** ***625,1939;*** ***1321,1916;*** ***1353,1165;*** ***664,1181***	

Note: Each coordinate pair must be specified on a new line with a semi-colon separator.

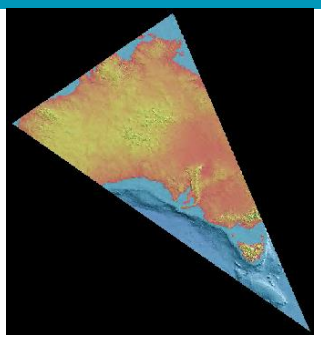
-shapefile vector.shp, layername, fid

Provide an Esri® Shapefile to define the non-Null or active area region defined by a closed polygon feature. Requirements:

- Shapefile projection must match the coordinate system of the input raster
- FID identifier must map to a polygon feature

-wktfile region.txt

Identical to “–worldregionfile” however in OGC® WKT format. Coordinates must match the dataset source projection.

Dataset coordinates	Output result
POLYGON((112.65148514850258721 -23.95247524751998469,135.48118811879865575 -8.2336633663325216,154.94257425741173506 -53.51881188118210986,112.65148514850258721 -23.95247524751998469))	

Advanced Options

-bitdepth 8 | 16

Force the output cell type to the provided value. If undefined the GeoCompressor will maintain the input celltype so this option is only required in instances where the input type is incorrectly detected. Int16 is supported in ECW v3 and JPEG2000 output.

Note: Setting –bitdepth to 8 (uint8) when the input range is uint16 will not rescale the data. GeoCompressor will merely truncate the data values and a warning will be logged.

-signed true | false

Used in conjunction with “–bitdepth 16” to set signed or unsigned int16 output. Signed INT16 is only supported in JPEG2000 output.

-colorspace greyscale | rgb | multiband

Both ECW and JPEG2000 formats support three colorspaces that are dependent on the band count. When undefined, GeoCompressor will maintain the colorspace of the input data. This option can be used in conjunction with –bandlist to subset multiband input to create RGB or Greyscale output.

-bandlist 0,1,2

Explicitly select which input bands are to be compressed, indexed from 0. Default option is to compress all bands in original order.

This option allows bands to be reordered such as BGR to RGB (eg. 2,1,0) or to be targeted selectively when used with the –colorspace option. For example a 7 band Multiband input can be mapped to a RGB 7,4,2 output by specifying –colorspace RGB –bandlist 7,4,2.

-logfile C:\log.txt

All information printed to the console will also be logged to file, useful for auditing or performance purposes.

-genstats true | false

When “true” the GeoCompressor will keep count of histogram bins and data statistics such as mean, mode, standard deviation for writing to the output file. This is calculated during the compression process and is only available when writing ECW v3 files (with embedded metadata support). For performance reasons the calculated statistics is based on the input pixels rather than the output compressed pixels. This trade-off will be improved in a future release.

Default value “true” for ECW v3.

-inputnodata (nodata value)

Set the “nodata” value that will be applied to all bands for each input file. This represents a value in the input file that will be treated as NULL value when compressing. E.g. “255” would be equivalent to white, “0” to black. This is useful when mosaicking images to remove black or white areas around the dataset (often caused by reprojection).

-nullvalue min | max

The colour definition to assign to null blocks when –opacity is set to 1 or 2. The min/max represents the smallest or largest pixel value for the specified bit depth range. For uint8, min will be 0 (black) and max, 255 (white). Default value “min”.

-interactive true | false

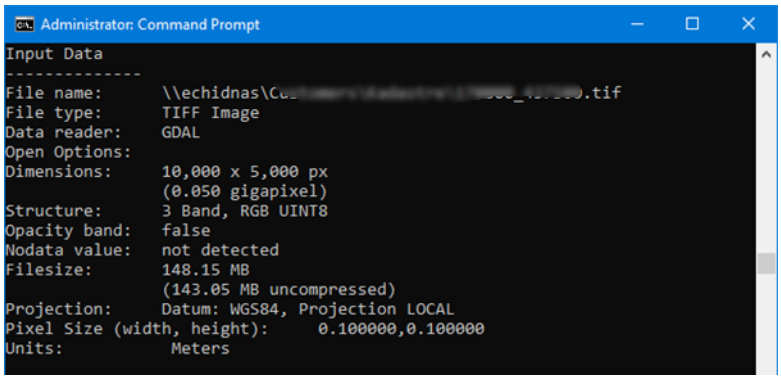
Compressor will prompt for user input when the following conditions are met,

- Insufficient memory cache
- Output file already exists

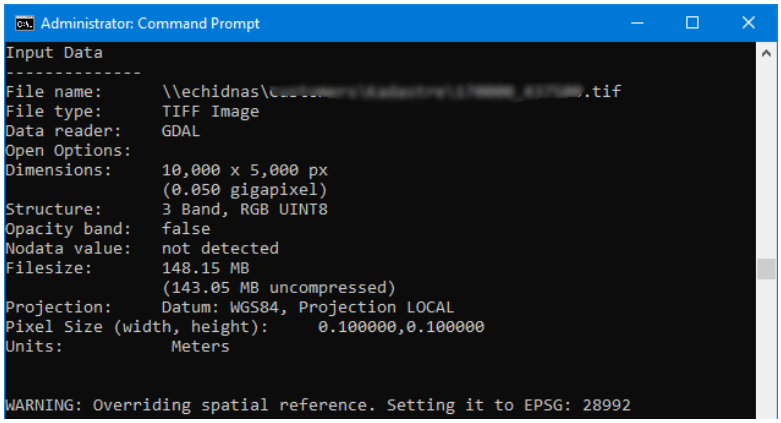
When using batch compression mode via the GUI, interactive defaults to “false”. Command line defaults to “true”.

-srs EPSG:2180

Override the input projection definition and use the specified EPSG code when writing the georeferencing information in the output file. This parameter should only be used where the Image Compressor is unable to detect the source projection correctly. For example, the following IMG input lists a projection code of WGS84 / LOCAL indicating an offset was detected but EPSG lookup failed.



Where input files are known to have an EPSG code however the code was not detected correctly, the srs value (-srs EPSG:28992) can be passed into the compressor to set this value explicitly and a relevant warning will be shown in the log.



Note: GeoCompressor will not perform any reprojection of the source to target projection systems. This option should not be used when the software is correctly listing the input projection code. For reprojection needs, ERDAS IMAGINE is recommended. For modifying existing ECW files, refer to ECW Header Editor Utility.

-xml (C:\data\mosaic.xml)

Define all compression and mosaic parameters within the XML project file and pass into the compressor. This option is exclusive of all others, including <input>, <output> and other options.

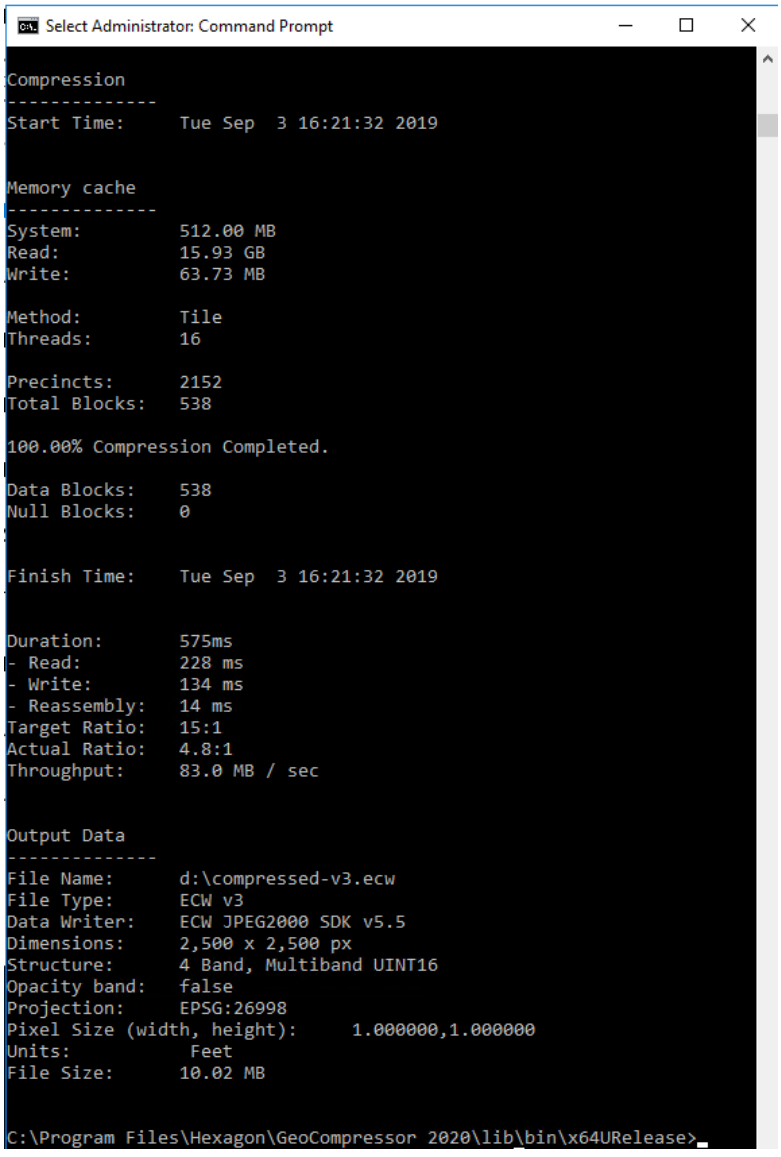
Example usage: ImageCompressor.exe –xml c:\data\mosaic.xml

Reporting Example

GeoCompressor records a variety of metrics of the input, output, hardware and throughput for each compression job. When reporting issues or concerns regarding processing speed always include the full report.

The information is also written to a logfile when used with the –logfile option, which includes additional information such as the timing for each percentage increment.

Figure 3 - Command line processing report

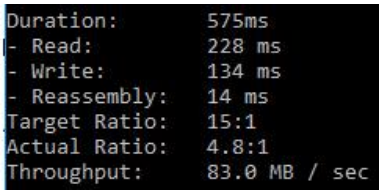


Key reporting elements include:

- Data Reader used (GDAL, ERMLib, ECWJP2 SDK)
- Size and structure of the input data
    - Internal block size (512x512) for tile, (10000x1) for strip

    - Compression used, if any (RLE, LZW, JPEG, Packbits)
    - Band count, cell type, opacity and projection information
    - Uncompressed filesize and gigapixels
- Calculated memory cache settings
- Output compression results in terms of throughput, time and actual compression rate achieved
- Duration is now broken down into sub-components to give further insights into the processing bottlenecks. The sample shows that the compression task took 9 minutes, 2 seconds to complete. Of interest is the three components where:
    - “Read” is the CPU time the ECWJP2 SDK compression threads are waiting for input
    - “Write” is the CPU time the ECWJP2 SDK compression threads are performing the DWT, writing to disk or in general term, performing the compression
    - “Reassembly” is the user time needed to reassemble the file at the end of a compression. This phase will always take the last 5% of each compression job and be associated with low CPU and high levels of disk I/O reading from the tempdir location and writing to the output location.

Figure 4 – Output timing breakdown



The above example indicates the majority of the time was waiting on data input. The actual compression (Write) was a small relative percentage of the CPU time, which is very important to differentiate. This could indicate slow or remote storage, poorly formatted input structure, bottlenecks in the mosaicking pipeline, threading issues in the data reader (GDAL) or all of the above.

An ideal target is for the Read/Write to be almost the same however depending on the situation this might be unattainable. The critical piece of information to highlight is that the ECW Compression pipeline is almost always bottlenecked by reading data from disk. Contact Support for advice or recommendations to improve throughput if speed is an important criteria for your business.

Note: Read + Write + Re-assembly Time will not equal the Duration. See https://www.gnu.org/software/libc/manual/html_node/Processor-And-CPU-Time.html

Usage Examples

GeoTIFF to ECW v2 using the line encoder

```
ImageCompressor c:\data\test.tif e:\data\test.ecw –method line
```

ArcInfo Binary Grid (AIG) to lossless JPEG2000 INT16

```
ImageCompressor c:\dem\w001000.adf e:\data\dem.jp2 –targetrate 1 –signed true
```

4 band BGRN UINT16 IMG HFA to RGB ECW v3 with target rate of 20:1 and band stats created

```
ImageCompressor c:\multi\bgrn.img e:\data\rgb.ecw –version 3 –bandlist 2,1,0 –targetrate 20 -genstats true
```

3 band ERMapper ALG to ECW v3 with null blocks and opacity

```
ImageCompressor c:\alg\mosaic.alg e:\data\mosaic.ecw –version 3 –opacity 2 –shapefile
c:\alg\region.shp,region,0
```

Mosaic XML project to ECW v2 at 20:1 target ratio (defined within the XML file)

```
ImageCompressor -xml project.xml
```

Panchromatic IMG to lossy JPEG2000 with opacity defined by WKT region

```
ImageCompressor c:\jp2\data.img e:\data\output.jp2 –wktfile c:\jp2\region.txt –opacity 0
```

Multiband 7 band TIF to RGB ECW with band selection

```
ImageCompressor multiband.tif rgb.ecw –bandlist 7,4,2 –colorspace rgb
```

ECW v3 Update Example

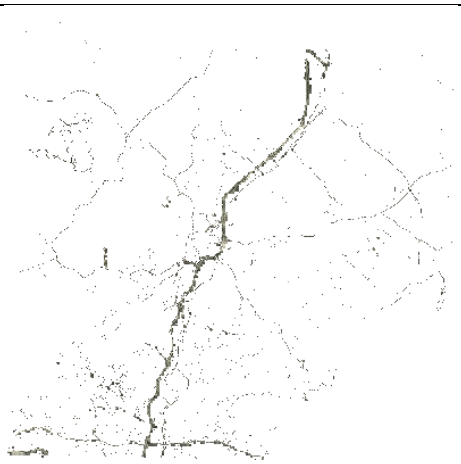
The following example shows an existing ECW v3 file being updated with 0.15% new data. Previously this use-case required the whole project to be recompressed however now with the selective algorithm it resolves a major limitation of the format in how customers could use ECW as a “living” format as new data was received. The larger the ECW, the greater the problem this limitation becomes. The speed performing the update is comparable to the speed compressing only the updated region and in this example only took 13 seconds.

When an ECW v3 file is updated, the previous data is inaccessible. End-users are equally unaware that the file has been updated as it behaves just like any other ECW file. All applications that can read ECW v3 can also read updated files.

Note: There is no restriction to the amount of new data that can be updated but if more than 30% of the original is to be updated it’s recommended to use the standard compression task rather than update due to processing overheads. Update is more suited for extremely large ECW files with update regions in the few percentage (in terms of coverage).

Source Data

File name: roads.ecw
File type: ECW Image
Dimensions: 243,536 x 247,155 px
(60.191 gigapixel)
Structure: 4 Band, RGB UINT8
Opacity band: true
Statistics: true
Filesize: 713.07 MB
(168.17 GB uncompressed)
Projection: EPSG:2240



Update data

File name: ECWupdate.xml
Dimensions: 25,000 x 5,000 px
(0.125 gigapixel)
Structure: 4 Band, RGB UINT8
(2 image files)
Opacity band: true
Filesize: 0.79 KB
(357.63 MB uncompressed)
Projection: EPSG:2240
Update data area: 0.149%

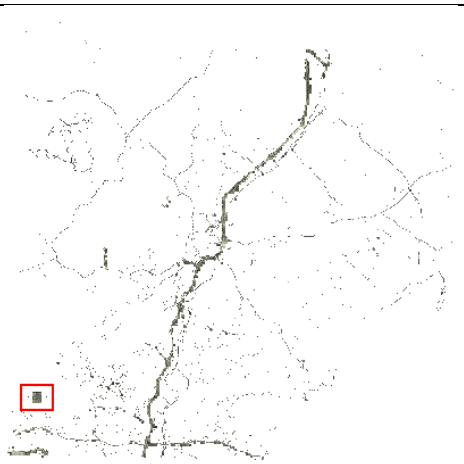


Figure 5 - Updated ECW V3 with highlighted addition

Result

Duration: 0 hours 0 mins 13 seconds
- Read: 8930 ms
- Write: 3035 ms
- Reassembly: 400 ms
Throughput: 36.6 MB / sec

Output Data

File Name: roads.ecw
File Type: ECW v3
Data Writer: ECW JPEG2000 SDK v5.2
Dimensions: 243,536 x 247,155 px
Structure: 4 Band, RGB UINT8
Opacity band: true
Projection: EPSG:2240
File Size: 717.57 MB

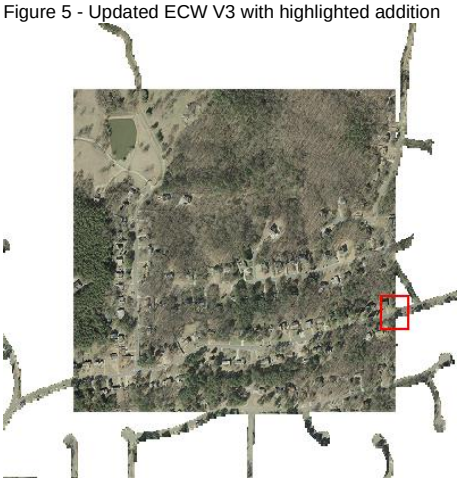


Figure 6 - Zoomed in view of updated area

Figure 7 – 1:1 Native resolution shows seamless addition on left with the original data on the right. Quality is unchanged.

XML Project File

GeoCompressor uses its own XML Project file to define image compression parameters and is the only way to define a mosaic or update input task. If using the GUI the wizard interfaces will create the XML for you as part of the wizard. The XML offers an easy integration point for third-party systems to define compression tasks in a programmatic way and pass in the XML to the command line tool using the –XML switch.

Note:

- Metadata and RPC elements are optional and are only relevant for ECW v3 output.
- A mosaic operation is implied when there is more than 1 input file listed while a batch operation is implied when there is only 1 input (operation=”create”).
- To update a region within an existing ECW V3 file, operation must be set to update (operation=”update”). The output file denotes the existing ECW v3 that you wish to update with the data listed as inputs. Multiple inputs are supported as GeoCompressor will mosaic and update on-the-fly.
- In most situations, GeoCompressor will default to an appropriate value otherwise it will exit when parsing the XML

```
<?xml version="1.0" ?>
<imagecompressor version="1.1">

  <!--compression task: attributes describe the options for the process-->
  <compresstask operation=”create|update”
                interactive="true" logfile="C:/temp/compressor.log"
                loglevel="Info" method="tile" tempdir="C:/temp"
                threads="4" memcache="0.25">

    <!--inputs: multiple files determine mosaic mode-->
    <inputs>
      <file bandlist="0,1,2" path="C:/dir/file_rgb1.ecw" zindex="0" nodata="0"/>
      <file bandlist="0,1,2" path="C:/dir/file_rgb2.ecw" zindex="1" nodata="0"/>
      <file bandlist="7,4,2" path="C:/dir/file_landsat5.ecw" zindex="2" nodata="255"/>
    </inputs>

    <!--output options-->
    <output path="C:/data/output/compressed.ecw" version="2">
      <!--output bandlist-->
      <bandlist>
        <band description="red" id="0"/>
        <band description="green" id="1"/>
        <band description="blue" id="2"/>
      </bandlist>

      <!--region definition from shapefile-->
      <region layerid="1" layername="file" name="region1"
              path="file.shp" type="shape"/>

      <!--region definition from WKT-->
      <region name="region2" path="file.wkt" type="wkt"/>
      <!--metadata: ecwv3 only-->
      <metadata>
        <item name="classification" value="raster image"/>
        <item name="acquistiondate" value="2013-09-12"/>
        <item name="acquisitionsensorname" value="Landsat 7"/>
        <item name="author" value="John Smith"/>
        <item name="copyright" value="Intergraph"/>
        <item name="company" value="Intergraph"/>
        <item name="email" value="compressor@imagery.net"/>
        <item name="address" value="2 Abbotsford St, West Leederville WA 6007"/>
        <item name="telephone" value="(08) 9388 2900"/>
      </metadata>

      <!—Embedded RPC: ECW v3 only-->
      <rpcdata>
        <errorbias>12.23</errorbias>
        <errorrandom>0.48</errorrandom>
        <lineoffset>3522</lineoffset>
        <sampleoffset>4406</sampleoffset>
        <latitudeoffset>35.2298</latitudeoffset>
        <longitudeoffset>-80.8601</longitudeoffset>
```

```
      <heightoffset>186</heightoffset>
      <linescale>3639</linescale>
      <samplescale>4421</samplescale>
      <latitudescale>0.0780</latitudescale>
      <longitudescale>0.1020</longitudescale>
      <heightscale>501</heightscale>

      <!---the below four coefficient need to have 20 comma separated parameters -->
      <linenumeratorcoefficients>
            -2.082755E-03,-2.790828E-02,...-5.147002E-07
      </linenumeratorcoefficients>
      <linedenominatorcoefficients>
            1.000000E+00,-3.544814E-05,...+2.449699E-07
      </linedenominatorcoefficients>
      <samplenumeratorcoefficients>
         -7.918007E-05,+1.000871E+00,...+6.943991E-08
      </samplenumeratorcoefficients>
      <sampledenominatorcoefficients>
      +1.000000E+00,-2.142645E-04,...+0.000000E+00
      </sampledenominatorcoefficients>
    </rpcdata>

     <!--options-->
     <options>
      <option name="bitdepth" value="8"/>
      <option name="targetrate" value="30"/>
      <option name="colorspace" value="rgb|greyscale|multiband"/>
      <option name="genstats" value="true"/>
      <option name="blocksizex" value="64"/>
      <option name="blocksizey" value="64"/>
      <option name="opacity" value="0|1|2|3"/>
      <option name="nullblocks" value="region2"/>
      <option name="nullvalue" value="min"/>
      <option name="qualitylayers" value="50"/>
      <option name="srs" value="EPSG:4326"/>

     </options>
    </output>
  </compresstask>
</imagecompressor>
```

Supported Input Formats

Hexagon Formats

- Enhanced Compressed Wavelet ECW (.ecw) version 2 and 3
- JPEG2000 (.jp2, .jpc, .j2k, .jpf, .j2c, or .jpx) files
- ERMapper Algorithm: .alg
- ERMapper Raster: .ers
- Terrashare Raster Backdrop file .rbd

Other Industry Formats

- MrSID : Multi-resolution Seamless Image Database (MrSID)
- VRT : Virtual Raster
- GTiff : GeoTIFF
- NITF : National Imagery Transmission Format
- RPFTOC : Raster Product Format TOC format
- ECRGTOC : ECRG TOC format
- HFA : Erdas Imagine Images (.img)
- SAR_CEOS : CEOS SAR Image
- CEOS : CEOS Image
- JAXAPALSAR : JAXA PALSAR Product Reader (Level 1.1/1.5)
- GFF : Ground-based SAR Applications Testbed File Format (.gff)
- ELAS : ELAS
- AIG : Arc/Info Binary Grid
- AAIGrid : Arc/Info ASCII Grid
- GRASSASCIIGrid : GRASS ASCII Grid
- SDTS : SDTS Raster
- DTED : DTED Elevation Raster
- PNG : Portable Network Graphics
- JPEG : JPEG JFIF
- MEM : In Memory Raster
- JDEM : Japanese DEM (.mem)
- GIF : Graphics Interchange Format (.gif)
- BIGGIF: Graphics Interchange Format (.gif)
- ESAT : Envisat Image Format
- BSB : Maptech BSB Nautical Charts
- XPM : X11 PixMap Format
- BMP : MS Windows Device Independent Bitmap
- DIMAP : SPOT DIMAP
- AirSAR : AirSAR Polarimetric Image
- RS2 : RadarSat 2 XML Product
- SAFE : Sentinel-1 SAR SAFE Product
- PCIDSK : PCIDSK Database File
- PCRaster : PCRaster Raster File
- ILWIS : ILWIS Raster Map
- SGI : SGI Image File Format 1.0
- EIR : Erdas Imagine Raw
- DIPEx : DIPEx
- LCP : FARSITE v.4 Landscape File (.lcp)
- GTX : NOAA Vertical Datum .GTX
- SRTMHGT : SRTMHGT File Format
- Leveller : Leveller heightfield
- Terragen : Terragen heightfield
- ISIS3 : USGS Astrogeology ISIS cube (Version 3)
- ISIS2 : USGS Astrogeology ISIS cube (Version 2)
- PDS : NASA Planetary Data System
- VICAR : MIPL VICAR file
- TIL : EarthWatch .TIL
- ERS : ERMapper .ers Labelled
- L1B : NOAA Polar Orbiter Level 1b Data Set
- FIT : FIT Image
- GRIB : GRIdded Binary (.grb)
- RMF : Raster Matrix Format
- WCS : OGC Web Coverage Service
- WMS : OGC Web Map Service
- MSGN : EUMETSAT Archive native (.nat)
- RST : Idrisi Raster A.1
- INGR : Intergraph Raster
- GSAG : Golden Software ASCII Grid (.grd)
- GSBG : Golden Software Binary Grid (.grd)
- GS7BG : Golden Software 7 Binary Grid (.grd)
- COSAR : COSAR Annotated Binary Matrix (TerraSAR-X)
- TSX : TerraSAR-X Product
- COASP : DRDC COASP SAR Processor Raster
- R : R Object Data Store
- MAP : OziExplorer .MAP
- KMLSUPEROVERLAY : Kml Super Overlay
- Rasterlite : Rasterlite
- MBTiles : MBTiles
- PLMOSAIC : Planet Labs Mosaics API
- CALS : CALS (Type 1)
- WMTS : OGC Web Mab Tile Service
- SENTINEL2 : Sentinel 2
- MRF : Meta Raster Format
- PNM : Portable Pixmap Format (netpbm)
- DOQ1 : USGS DOQ (Old Style)
- DOQ2 : USGS DOQ (New Style)
- GenBin : Generic Binary (.hdr Labelled)
- PAux : PCI .aux Labelled
- MFF : Vexcel MFF Raster
- MFF2 : Vexcel MFF2 (HKV) Raster
- FujiBAS : Fuji BAS Scanner Image
- GSC : GSC Geogrid
- FAST : EOSAT FAST Format
- BT : VTP .bt (Binary Terrain) 1.3 Format
- LAN : Erdas .LAN/.GIS
- CPG : Convair PolGASP
- IDA : Image Data and Analysis
- NDF : NLAPS Data Format

- LOSLAS : NADCON .los/.las Datum Grid Shift
- NTv2 : NTv2 Datum Grid Shift
- CTable2 : CTable2 Datum Grid Shift
- ACE2 : ACE2
- SNODAS : Snow Data Assimilation System
- KRO : KOLOR Raw
- ROI_PAC : ROI_PAC raster
- ENVI : ENVI .hdr Labelled
- EHdr : ESRI .hdr Labelled
- ISCE : ISCE raster
- ARG : Azavea Raster Grid format
- RIK : Swedish Grid RIK (.rik)
- USGSDEM : USGS Optional ASCII DEM (and CDED)
- GXF : GeoSoft Grid Exchange Format
- PLSCENES : Planet Labs Scenes API
- HTTP : HTTP Fetching Wrapper
- NWT_GRD : Northwood Numeric Grid Format .grd/.tab
- NWT_GRC : Northwood Classified Grid Format .grc/.tab
- ADRG : ARC Digitized Raster Graphics
- SRP : Standard Raster Product (ASRP/USRP)
- BLX : Magellan topo (.blx)
- SAGA : SAGA GIS Binary Grid (.sdat)
- XYZ : ASCII Gridded XYZ
- HF2 : HF2/HFZ heightfield raster
- OZI : OziExplorer Image File
- CTG : USGS LULC Composite Theme Grid
- E00GRID : Arc/Info Export E00 GRID
- ZMap: ZMap Plus Grid
- NGSGEOID : NOAA NGS Geoid Height Grids
- IRIS : IRIS data (.PPI, .CAPPi etc)
- DB2ODBC : IBM DB2 Spatial Database
- GPKG : GeoPackage

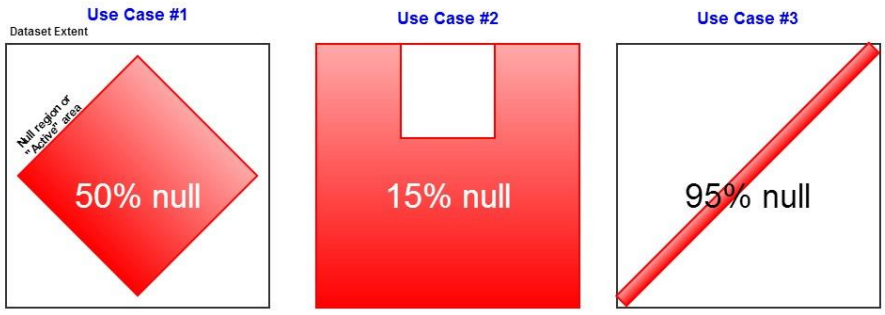
Note: When mosaicking input data, not all GDAL supported formats listed are available due to non-thread safe operations. The list above is supported for batch processing or single input only. If random errors occur when processing these input format types in tile mode, setting –method to line is recommended.

Null Block Analysis

A major improvement with ECW v3 is the introduction of null blocks that can offer further file storage savings and compression performance compared with ECW v2 or JPEG2000 without sacrificing image quality. The key criteria as to whether null blocks should be enabled is the relationship of the input data extent to the amount of null or no-data areas and the size of the input image. Generally, the higher amount of null area defined by the input region with increasing image input size, the greater the gains that enabling null blocks will provide. This feature is particularly suited to imagery corridor projects such as rivers, roads, transmission lines, pipelines, railways, or large national mapping projects.

The compressor output lists two important region metrics to help identify suitability:

1. Number of vertices – indicates the complexity of the input region. The greater the vertices, the more expensive spatial intersection tests will be. The compressor has implemented a variety of optimizations but where possible a simplified region will ensure the fastest throughput. The compressor has been tested with polygons with tens of thousands of vertices however it's expected most use-cases will on average only require vertices in the hundreds if not less.
2. Ratio of null area to data which can be visualized in the following diagrams where the red area is the region passed into the compressor and the white is the remaining area to be tagged as "null blocks"



An important observation in these examples are the first and third use-cases. Both have 4 vertices however clearly the percentage ratio to data is a lot higher in the third example at 95%. Therefore null blocks will provide the greatest benefit to the third image both in terms of additional file storage savings and compression speed. The first example will still benefit and is still a good candidate for null blocks however will not see as significant gains. Null blocks in this way will always provide varying levels of optimizations depending on input as highlighted in the following examples.

Comparison

Sydney Landsat scene

Input image

Dimensions: 15,221 x 14,661 px
(0.223 gigapixel)
Structure: 3 Band, RGB UINT8
Opacity band: false
Projection: EPSG:32656

Shapefile region (yellow)

Null Vertices: 5
Ratio to data: 30.234%

Null blocks enabled	Null blocks disabled
Hardware	**Hardware**
Platform: Windows 7 / Server 2008 R2	Platform: Windows 7 / Server 2008 R2
CPU Model: Intel(R) Core(TM) i7 CPU Q 740 @ 1.73GHz	CPU Model: Intel(R) Core(TM) i7 CPU Q 740 @ 1.73GHz
CPU Cores: 8	CPU Cores: 8
Memory: 8,128.00 MB	Memory: 8,128.00 MB
Memory cache	**Memory cache**
System: 512.00 MB	System: 512.00 MB
Read: 1,911.85 MB	Read: 1,911.85 MB
Write: 120.15 MB	Write: 120.15 MB
Threads: 8	Threads: 8
Precincts: 73376	Precincts: 73376
Total Blocks: 18344	Total Blocks: 18344
Data Blocks: 13251	Data Blocks: 18344
Null Blocks: 5093	Null Blocks: 0
Duration: 0 hours 0 mins 55 seconds	Duration: 0 hours 1 mins 20 seconds
Target Ratio: 15:1	Target Ratio: 15:1
Actual Ratio: 31.3:1	Actual Ratio: 30.7:1
Throughput: 11.5 MB / sec	Throughput: 8.0 MB / sec

Output Data
File Name: f:\landsat-null.ecw
File Type: ECW v3
Data Writer: ECW JPEG2000 SDK v5.2
Dimensions: 15,221 x 14,661 px
Structure: 4 Band, RGB UINT8
File Size: 20.39 MB

Output Data
File Name: f:\landsat-no-null.ecw
File Type: ECW v3
Data Writer: ECW JPEG2000 SDK v5.2
Dimensions: 15,221 x 14,661 px
Structure: 4 Band, RGB UINT8
File Size: 20.79 MB

1. File savings: 400kb (~ 2% smaller)
2. Time savings: 25 seconds (~ 30% faster)

The small file size difference is expected in this example despite the 30% ratio to data because the input image is only small at 0.2 gigapixels. This means there are fewer resolution levels within the ECW file, which in turn means there are fewer null blocks in the output. Irrespective of the small file size improvement, enabling null blocks increases compression speed by 30%, which can be significant depending on use-case, for example compressing thousands of images in batch.

Note: It is a coincidence that the compression speed improvement of 30% matches the input region ratio to data of 30%. The exact performance throughput gains will vary.

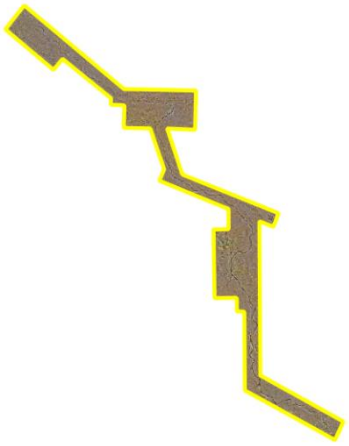
Corridor Mapping Example

Input image

Dimensions: 372,535 x 477,806 px
(177.999 gigapixel)
Structure: 4 Band, RGB UINT8
Opacity band: true
Projection: EPSG:28350

Shapefile region (yellow)

Null Vertices: 33
Ratio to data: 89.731%



Null blocks enabled	Null blocks disabled
Hardware Platform: Windows 7 / Server 2008 R2 CPU Model: Intel(R) Xeon(R) CPU E5410 @ 2.33GHz CPU Cores: 8 Memory: 16,380.00 MB	**Hardware** Platform: Windows 7 / Server 2008 R2 CPU Model: Intel(R) Xeon(R) CPU E5410 @ 2.33GHz CPU Cores: 8 Memory: 16,380.00 MB
Memory cache System: 512.00 MB Read: 2,319.47 MB Write: 1,775.53 MB	**Memory cache** System: 512.00 MB Read: 2,319.47 MB Write: 1,775.53 MB
Threads: 8	Threads: 8
Precincts: 57967752 Total Blocks: 14491938	Precincts: 57967752 Total Blocks: 14491938
Data Blocks: 1506776 Null Blocks: 12985162	Data Blocks: 14491938 Null Blocks: 0
Duration: 0 hours 53 mins 45 seconds Target Ratio: 15:1 Actual Ratio: 209.4:1 Throughput: 210.6 MB / sec	Duration: 3 hours 41 mins 19 seconds Target Ratio: 15:1 Actual Ratio: 169.3:1 Throughput: 51.2 MB / sec
Output Data File Name: g:\corridor-null.ecw File Type: ECW v3 Data Writer: ECW JPEG2000 SDK v5.2 Dimensions: 372,535 x 477,806 px Structure: 4 Band, RGB UINT8 File Size: 3,242.55 MB	**Output Data** File Name: g:\corridor-no-null.ecw File Type: ECW v3 Data Writer: ECW JPEG2000 SDK v5.2 Dimensions: 372,535 x 477,806 px Structure: 4 Band, RGB UINT8 File Size: 4,011.66 MB

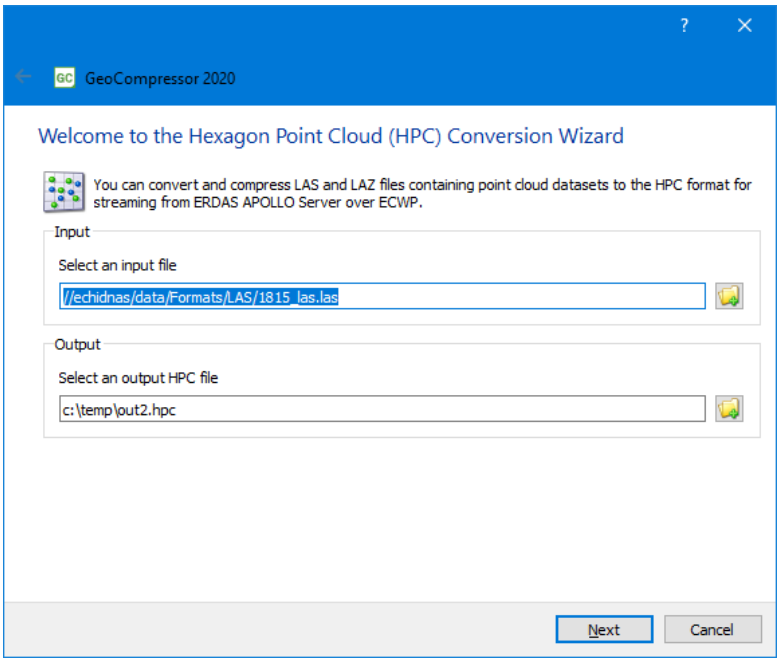
1. File savings: 769mb (~ 19% smaller)
2. Time savings: 2 hours 48 minutes (~ 410% faster)

This example shows the strengths of enabling null blocks. It has a relatively simple input region and a high level of null data of 89%. Unlike the previous example, we can now observe significant gains to both compression speed and file savings with no degradation to image quality. Although not tested here, decompression speed will also be improved.

Point Compressor Usage

Wizard Mode

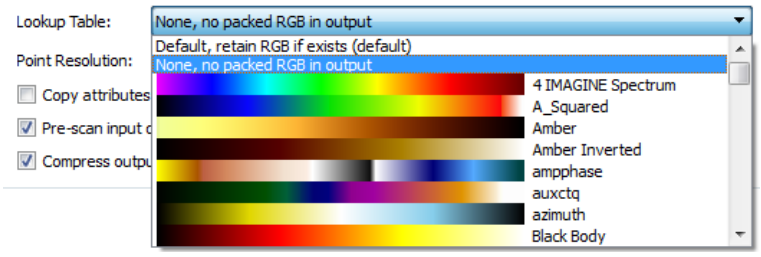
1. HPC Task wizard only supports single LAS or LAZ file inputs. Mosaicking or merging is not currently supported.



2. The next page confirms the available conversion settings.
    a. “Temp Directory” is the location that intermediate files are written to during conversion. They are cleaned up on completion of each task.
    b. “SRS Code” enables manual over-ride of the input EPSG code. By default, leaving this field empty will retain the input LAS/LAZ georeferencing. This should only be used where the EPSG lookup fails.
        i. Note: Setting this does not imply the points will be reprojected from the source to target systems, it’s informational only.
    c. “Lookup Table” presents a list of standard color tables that can be used to burn into the output HPC for visualization purposes only and is designed for datasets who do not have existing RGB values. Default behaviour is to retain existing RGB values and in the event none is found the “USGS Elevation” lookup will be used.
    d. “Point Resolution” allows thinning of the input data. Default value of 0 indicates the Compressor will use the same resolution reported as the Scale Factor in the LAS specification to ensure the output HPC retains the same number of points as the input. It is not recommended to specify a different value unless point thinning is intended.
    e. “Copy Attributes to Output File” when enabled will copy all Point Attributes stored in the LAS or LAZ input into the HPC output. This will increase the size of the HPC file. For visualization only of RGB values, stripping the attributes is recommended. Default is enabled.
    f. “Pre-scan input data to determine RGB color range” will scan all points to determine the color range when the input has RGB values greater than 0-255. This is required to ensure the HPC RGB values are not skewed incorrectly where the color range is a subset of uint16 values. Default true.
    g. “Compress output” can significantly alter the size of the HPC file. Default is enabled, however if intended to be used in other Hexagon software such as ERDAS IMAGINE where point’s may be edited you must

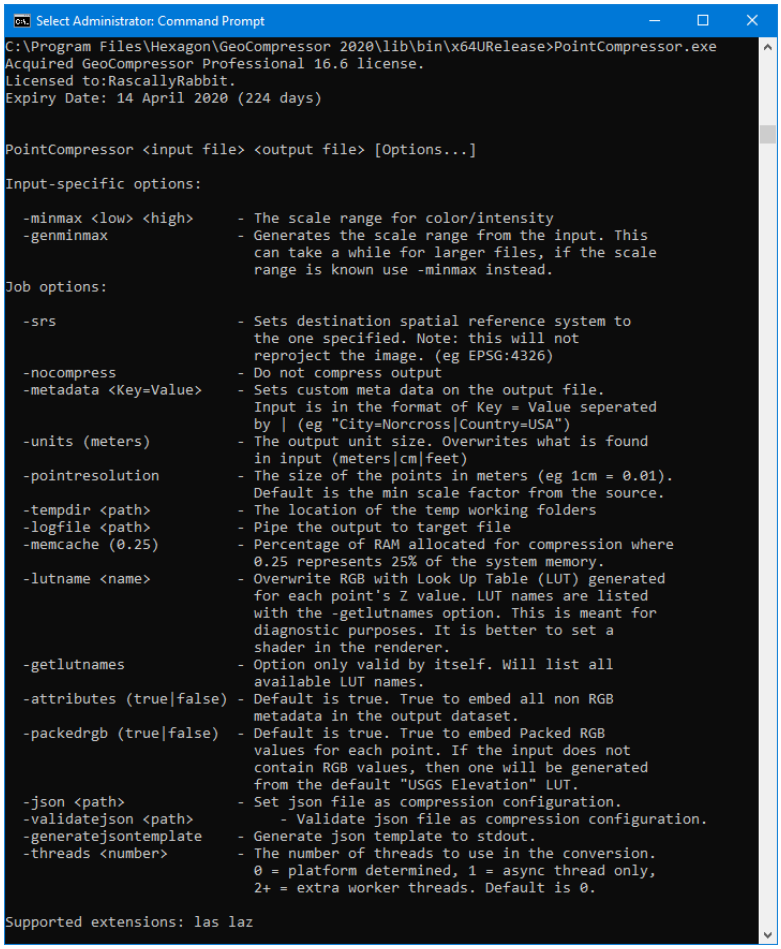
disable compression. For visualization usage or streaming via ERDAS APOLLO, retaining compression is recommended for performance and storage reasons. Default true.

Figure 8 - HPC Wizard output options



Command-Line Mode

The Point Compressor, like its Image Compressor sibling, is a transcoder of point cloud information. Its main use case is for the conversion to the Hexagon Point Cloud (HPC) format used across other Hexagon Geospatial software.



Input

Point Compressor supports .LAS (v1.1, 1.2, 1.3) and .LAZ input format types only. LAS v1.4 is not currently supported, nor is LAS, LAZ files with wave packed data.

Output

PointCompressor writes out *.HPC (v1.4.1) format and the extension must be provided.

Options

-minmax <low> <high>

The scale range for color intensity values if known.

-genminmax

Reads all input points to determine valid color range from the input. Depending on the input file size this may take a few minutes to complete. Not required if –minmax values are used.

-nocompress

Do not compress the output HPC file. Used only where attributes or points may be edited post-creation. If HPC is to be used for visualization or reading only, compression should always be enabled by omitting this option.

-units (meters|cm|feet)

Override the output unit size. By default the units will match the input file so this option should only be used if the automatic lookup fails. It will not convert between units.

-pointresolution (0.02)

The size of the points in units (meters). By default Pointcompressor will use the minimum Scale Factor reported by the LAS/LAZ Header information. Increasing this value will thin the output HPC but lose data.

-tempdir <path>

The storage location for intermediate files creating during compression. Defaults to the system %TEMP% environment. All files created during the task will be cleaned up on completion.

-logfile <path>

Write processing information to file as well as stdout.

-memcache (0.25)

Percentage of total RAM allocated for compression where 0.25 represents 25% of the System Memory. Default’s to 0.25.

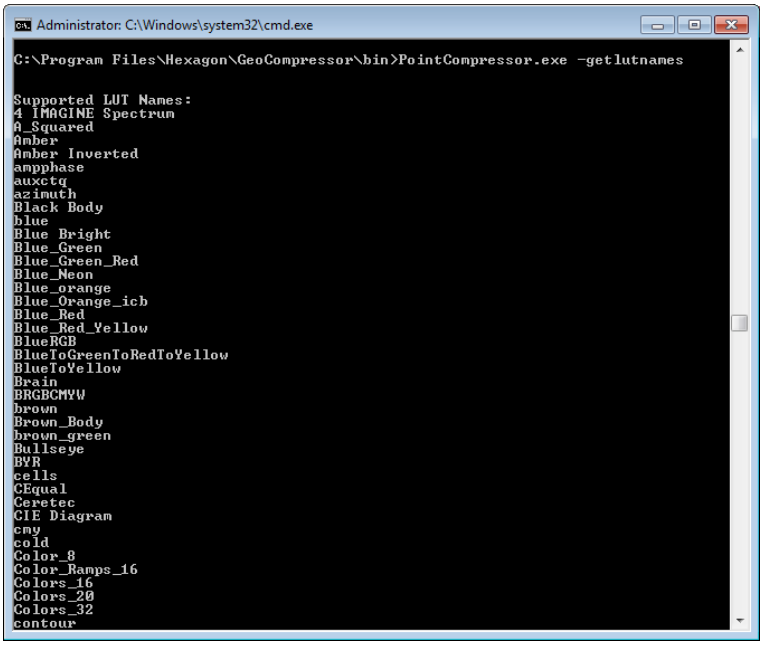
-lutname <name>

Used to define a Lookup Table to burn RGB values based on Elevation into the output HPC file. Default name is “USGS Elevation”, others can be looked up using the –getlutnames option below.

-getlutnames

Retrieves the list of supported lookup tables that can be passed in with –lutname. The list is built from all *.lut files stored in /lib/etc/erm/lut/. LUT files can be viewed in the GeoCompressor wizard interface or ERDAS IMAGINE.

Figure 9 - Supported Lookup tables



-attributes (true|false)

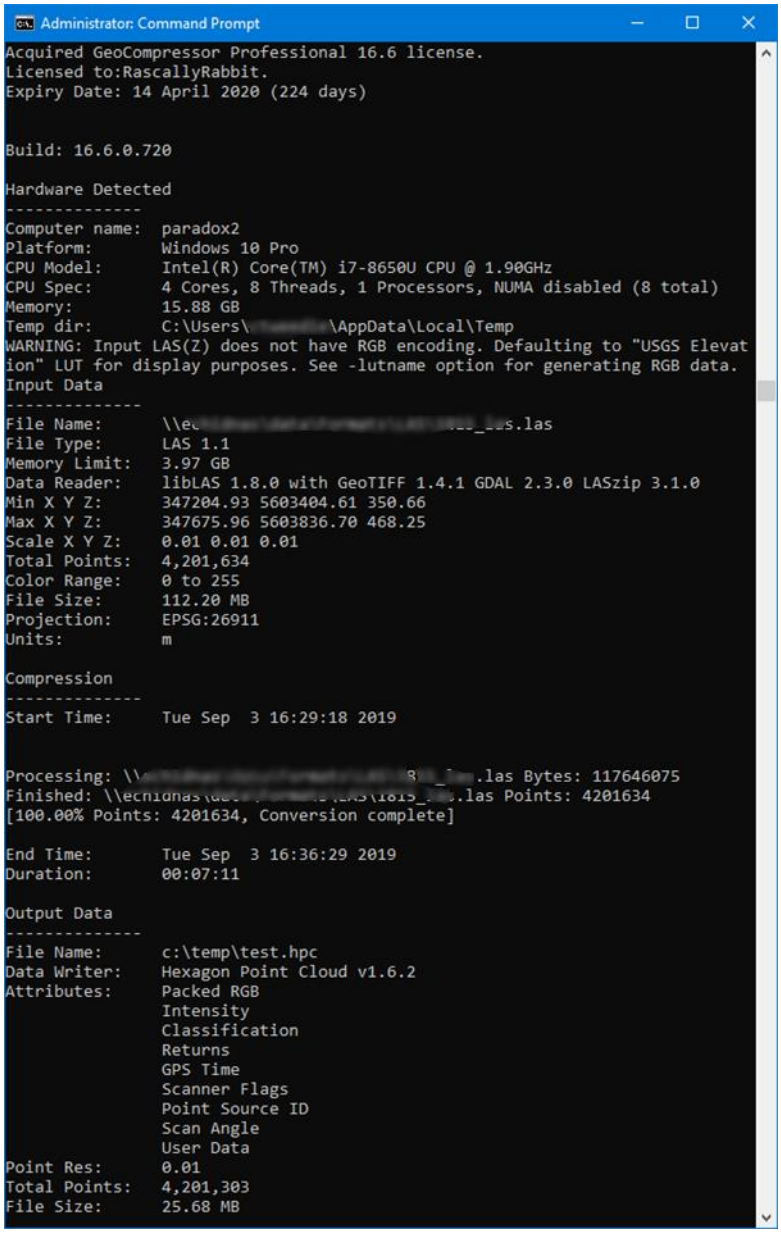
Defines whether the input point attributes are copied to the output HPC. Default is true. All metadata domains are retained and reported in the compressor output report (see below). When false, only RGB point attributes are written.

-threads <number>

Defines the number of threads to use when converting the input file. 0 calculates the number of threads automatically dependent on the number of processor cores in the system, 1 will use 1 asynchronous processing thread, 2 or more will be the extra number of asynchronous worker threads. The default is 0 (automatic).

Reporting Example

PointCompressor mimics the same reporting structure as ImageCompressor. The following example shows the output report with default options. Conversion to HPC completed in 9minutes and creates a 485MB file.



Usage Examples

Based on the previous LAS input file the following workflow describes the relationship between each processing option and the output HPC files. As HPC is voxel based, the point resolution value can significantly alter the HPC output.

Create a HPC for visualization purposes only with no point attributes

PointCompressor.exe c:\temp\RGB_5703374_5703375_0.las c:\temp\RGB_5703374_5703375_0.hpc –attributes false

```
Duration:        00:06:05

Output Data
---------------
File Name:       c:\temp\RGB_5703374_5703375_0-noattrib.hpc
Data Writer:     Hexagon Point Cloud v1.4.1
Attributes:      Packed RGB
Point Res:       0.000001
Total Points:    37,338,611
File Size:       305.73 MB
```

- 30% faster throughput, 30% smaller output size

Increase the default pointresolution as the reported LAS header value of 0.00001 is unusually small

PointCompressor.exe c:\temp\RGB_5703374_5703375_0.las c:\temp\RGB_5703374_5703375_0.hpc –attributes false –pointresolution 0.0001

```
Duration:        00:04:05

Output Data
---------------
File Name:       c:\temp\RGB_5703374_5703375_0-noattrib-0001.hpc
Data Writer:     Hexagon Point Cloud v1.4.1
Attributes:      Packed RGB
Point Res:       0.0001
Total Points:    37,338,611
File Size:       206.38 MB
```

- Further throughput gains and file size reductions
- Of note is the output point count is unchanged from the input (37,388,611 points) so data is not lost

Increase the default pointresolution to 0.01 (1cm)

PointCompressor.exe c:\temp\RGB_5703374_5703375_0.las c:\temp\RGB_5703374_5703375_01.hpc –attributes false –pointresolution 0.01

```
Duration:        00:01:51

Output Data
---------------
File Name:       c:\temp\RGB_5703374_5703375_0-noattrib-01.hpc
Data Writer:     Hexagon Point Cloud v1.4.1
Attributes:      Packed RGB
Point Res:       0.01
Total Points:    37,338,611
File Size:       110.82 MB
```

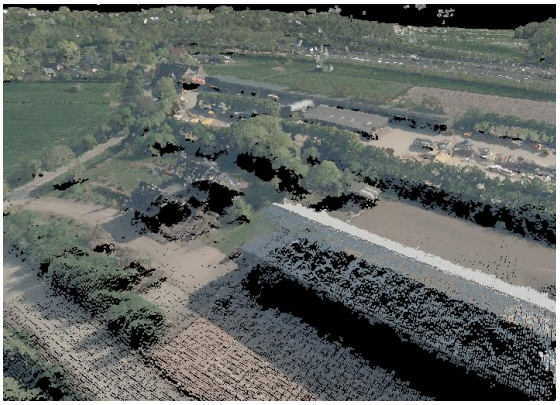
- Same benefits as previous with respect to time and filesize
- Output point count remains the same as input

Increase the default pointresolution to 0.5 (50cm)

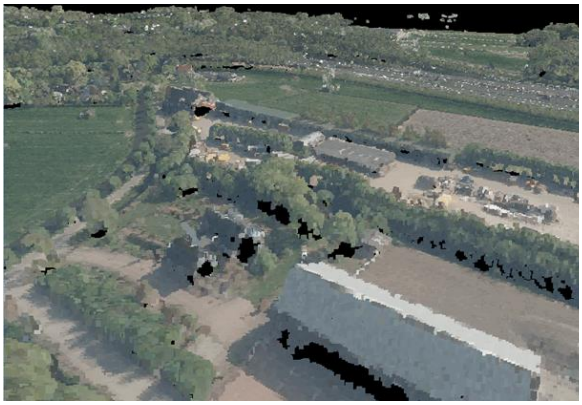
PointCompressor.exe c:\temp\RGB_5703374_5703375_0.las c:\temp\RGB_5703374_5703375_50.hpc –attributes false –pointresolution 0.5

```
Duration:        00:00:35

Output Data
---------------
File Name:       c:\temp\RGB_5703374_5703375_0-noattrib-01.hpc
Data Writer:     Hexagon Point Cloud v1.4.1
Attributes:      Packed RGB
Point Res:       0.5
Total Points:    11,140,724
File Size:       11.67 MB
```

- Now the output point count is less than the input. Any points within 0.5m would have been dropped, creating a file almost 10x smaller however containing 3x less points than the original.



- 1cm output HPC



- 50cm output HPC (lossy)

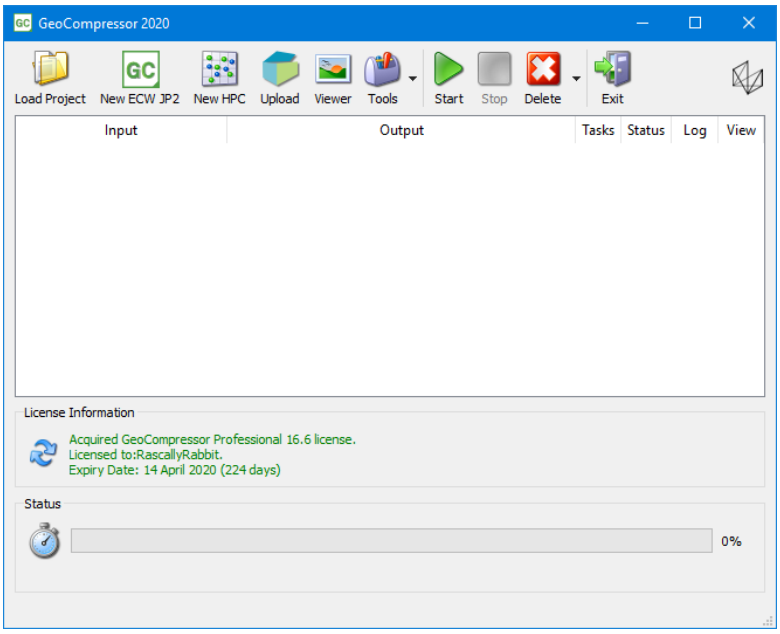
Upload Usage

Hexagon Smart M.App is a simple to use platform any organization or freelance developer can use to build lightweight and dynamic applications targeted to solve a specific problem. It combines content, business workflows, and geoprocessing into a single application to produce powerful visualizations. The primary focus of a Hexagon Smart M.App is to present users with analytical views of what was, what is, what could be, what should be and what will be. As part of the platform, the M.App Chest application allows users to manage online their content hosting and delivery service of data.

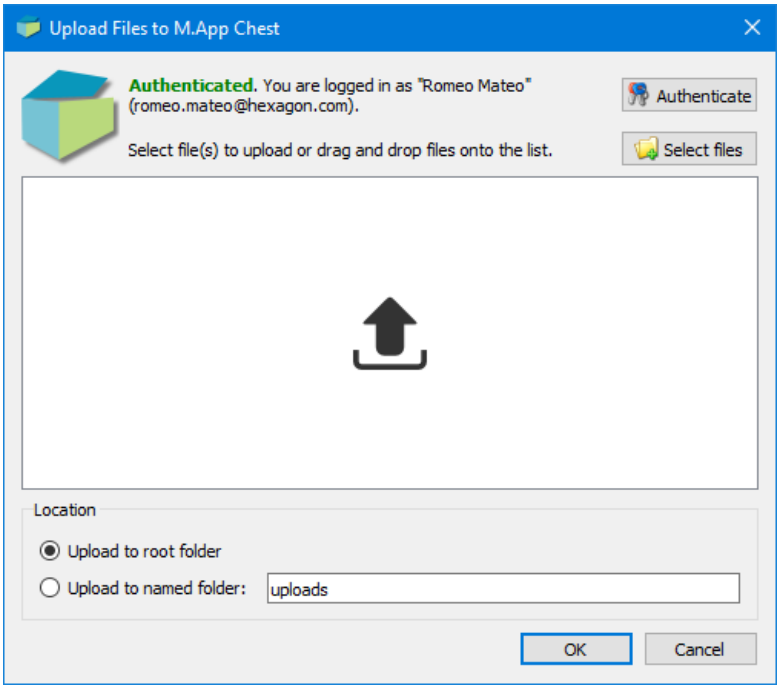
GeoCompressor now includes the ability to upload content to M.App Chest directly. A user must have a valid subscription to Smart M.App in order to use the upload capability. Although a user can upload data via a web browser from their account, the GeoCompressor upload feature allows uploads larger the 5GB and also features multiple concurrent HTTP connections for faster uploads.

GUI Mode

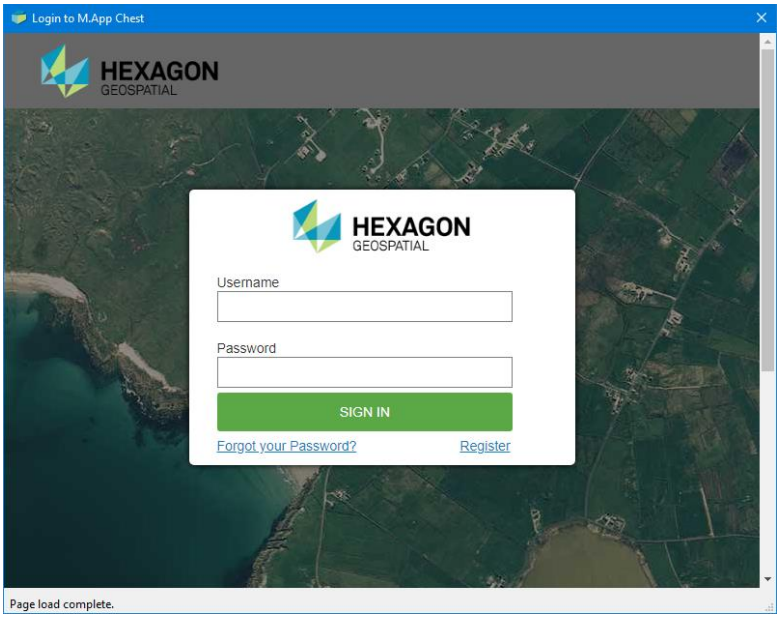
1. On the main window for the GeoCompressor click the “ Upload” button to open the upload window.



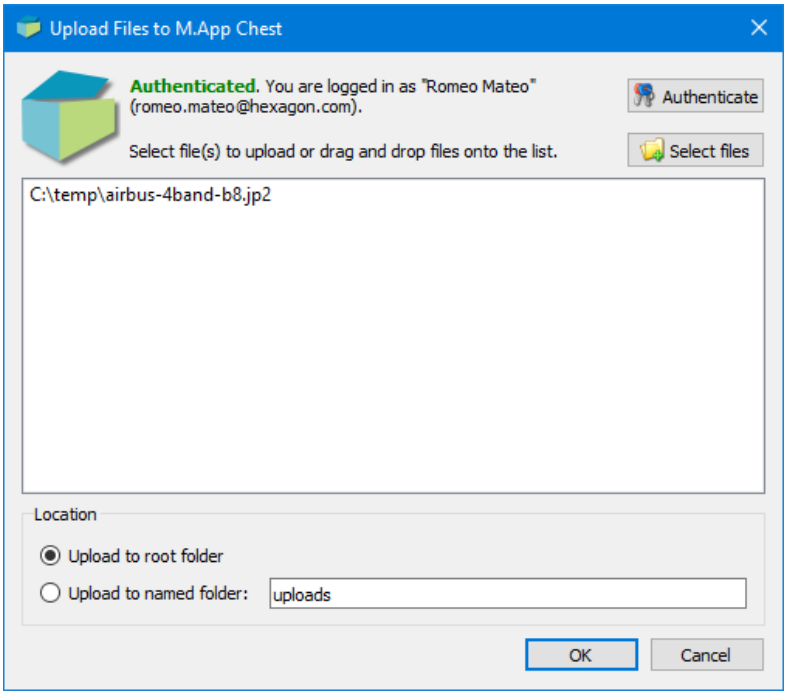
2. Enter your credentials and click “Authenticate”.



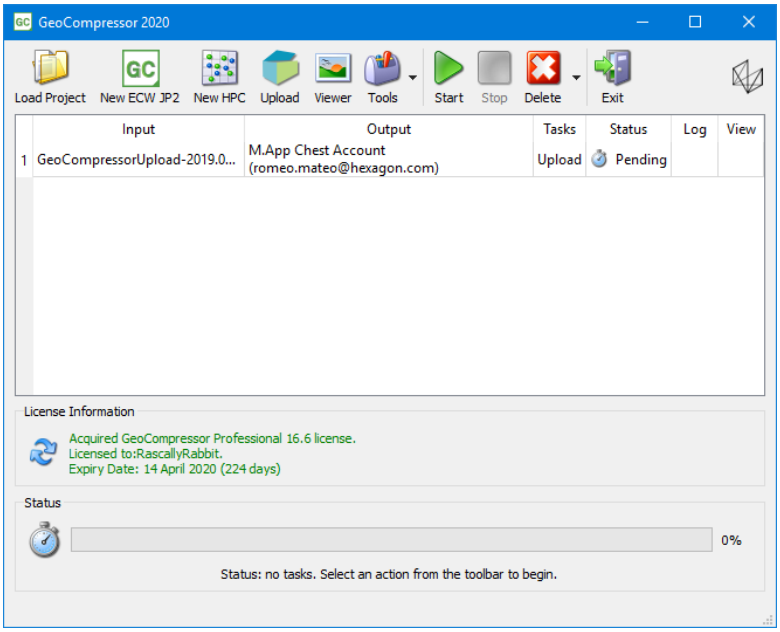
3. Enter your credentials into the Email and Password fields, then click “Log In” to finalize your access authorization.



4. The login window will close and you should now see the “Authenticated” message (green). Drag and drop files onto the upload panel from your desktop, or click the “Select files” button to open a file chooser to select files manually. Multiple files can be added to the window and queued for upload. In the “Location” panel, select either the root folder, or specify another folder name to upload data into. Click OK to continue.



5. The files are queued for processing, along with any other conversion tasks that were already queued. Click the start button to begin uploading files.



6. When the upload is complete, you can click the log file button to view the log file, or to view the file online in your M.App Chest account, click the view button.
7. When compressing an image, on the Summary page of Compression Wizard, in “Upload Output to M.App Chest” panel, select the check box to enable automatic upload once the compression is complete, then click “Authenticate” to login in to your Smart M.App account (follow the same procedure in step 3 above).
8. When the wizard is closed, the task appears as a normal compression task ready for processing. Upload will occur when the compression is complete.

Command-Line Mode

The functionality to upload data files to a Smart M.App Chest account has been supplied as a separate command-line application, so that users can script the upload of files in their own workflows. The applications MAppUploader.exe can be found in the bin directory of the GeoCompressor installation.

The format of the command is:

```
MAppUploader.exe inputfile <bearer-token> [options]
```

Input File

The input file is any file on the local file system that is supported by the Smart M.App environment (raster, vector etc). Alternatively, the file can be an XML file describing a set of files and the upload parameters. If an Xml file is specified, other options on the command line are ignored.

Sample input XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
  <mappuploader version="1.0">
    <uploadtask logfile="c:/temp/logfile.log">
    <inputs>
      <file path="C:\Data\compress\perth.tif" />
      <file path="C:\Data\compress\perth.rrd" />
      <file path="C:\Data\compress\perth.shp" />
      <file path="C:\Data\compress\perth.dbf" />
      <file path="C:\Data\compress\perth.shx" />
      <file path="C:\Data\compress\perth.prj" />
    </inputs>
    <output type="folder|catalogid"
        value="foldername|831f8055-0c4f-4ba6-81fd-e08769777a4f"
        description="Smart M.App Chest Account [first.last@comany.com]">
    </output>
    </uploadtask>
</mappuploader>
```

For the <output> element the type can be either “folder” or “catalogid”, in which case the value is a folder name or a folder catalog Id. If a name is specified, it must exist in the ROOT of the account, or else it will be created there. Nested folders are currently not supported. The attribute “description” is informational only and purely for displaying in the output colum of the task table.

For the example, the catalog will identify the attachments for the primary file (e.g. the dbf ,shx, prj for the shape file and the prj and rrd for the tiff file) and catalog them accordingly when they are grouped into one upload.

Bearer Token

The bearer token for the user’s current authenticated session. This must be acquired separately, either through a browser login to M.App Chest, or using the authentication workflow APIs provided by Smart M.App Foundation. For more details refer to the Smart M.App documentation.

Options

-environment <mapp|staging|development>

The environment to upload the data to, mapp (production) staging or development environments. The default is “mapp”. For most customers, only the production environment is relevant, staging and development is for

partners and application developers who are testing their data on pre-release versions of the Smart M.App platform. Note that the bearer token must be generated in the same environment as the upload target.

-foldername <name>

The name of the folder in the user's M.App Chest account to upload the data to. The default is the "ROOT" folder. If both folderid and foldername are specified, the folder Id is used.

-folderid <Id>

The catalog Id of the folder in the user's M.App Chest account to upload the data to. The default is the "ROOT" folder. If both folderid and foldername are specified, the folder Id is used.

-logfile <filename>

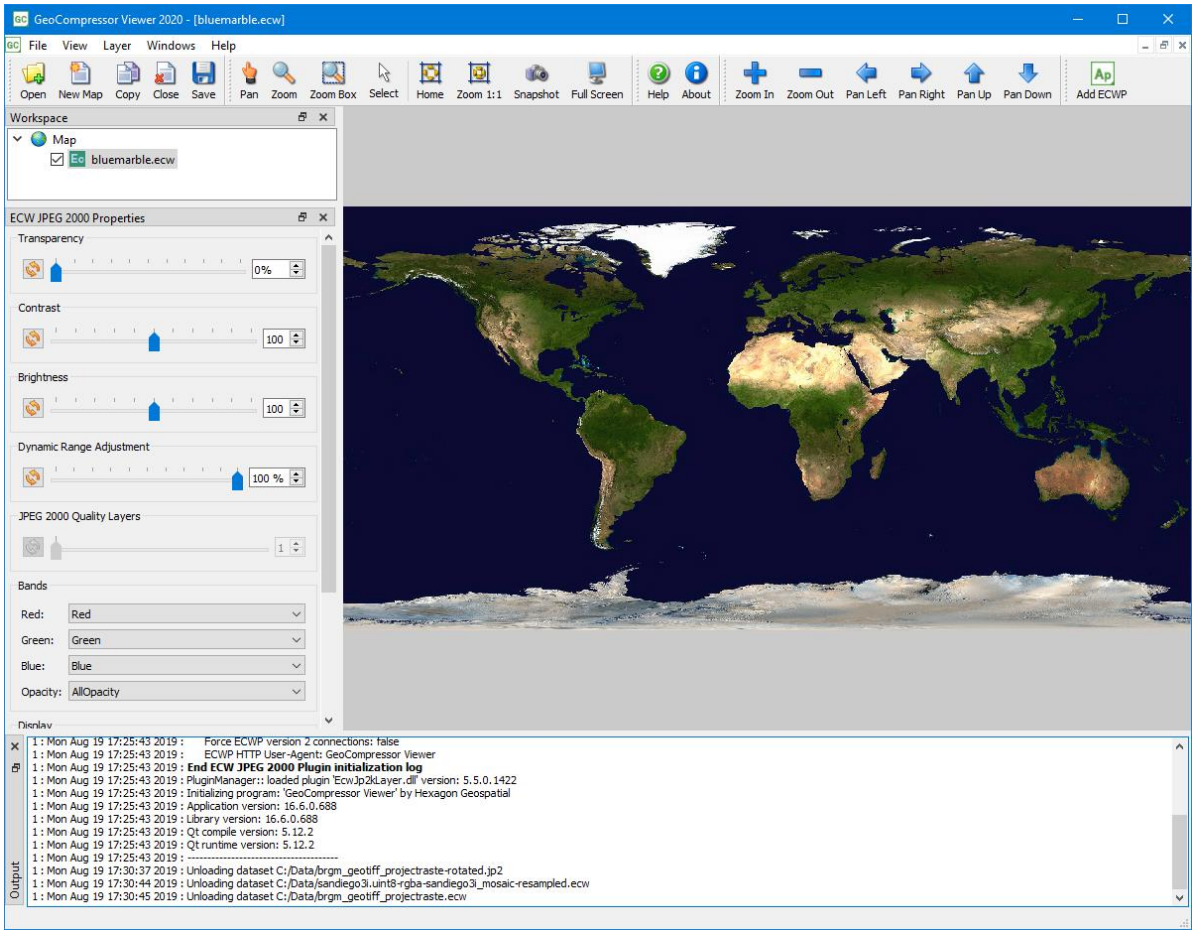
The filename to write output to. This is useful when diagnosing issues with the upload process.

GeoCompressor Viewer Usage

Overview

GeoCompressor Viewer application is a simple ECW and JPEG 2000 image viewer. It can also view imagery over the internet via the ECWP protocol from an ERDAS APOLLO Server. This component does not require a GeoCompressor license to use.

The application is an MDI (multiple document interface) application that allows you to pan and zoom around very large geospatial imagery. You can add multiple layers (for images in the same projection).



Toolbar Functions

Standard Toolbar

- Open – open an image file from local the machine.
- New Map – create an empty map window.
- Copy – copy the current map window to a new map window.
- Close – close the current map window.
- Save – save the current map window, including extents and properties, to an XML file, which can be loaded later back into a new map window.

View Toolbar

- Pan – pan the image up, down, left or right by clicking the left mouse button and dragging across the map. Right click will instigate interactive zoom node (see below).
- Zoom – interactively zoom in or out of the image by clicking the left mouse button and dragging down or up respectively.
- Zoom Box – zoom to a region of interest by dragging a rectangle on the map window.
- Select – select a feature or profile the image by clicking the left mouse button and dragging across the map. The current map SRS and cursor coordinates will display in the lower left corner of the application status bar.

IHome – zoom to the full extents of all layers.

- Zoom 1:1 – zoom to the 1:1 pixel resolution (1 pixel on screen equals 1 pixel in the source image) of the currently selected raster layer. If no layer is selected, the first raster layer found will be used.
- Snapshot – Save a screen snapshot in PNG format of the current map window to the user's desktop.
- Full Screen – put the viewer into full screen mode. This maximises the application so that no borders are present and takes over the entire screen. This allows the user to maximise the amount of imagery that can be displayed in the map window.

Navigation Toolbar

- Zoom In, Zoom Out – zoom in or out of the map, incrementally with each click of the button.
- Pan Left, Pan Right, Pan up, Pan Down – pan incrementally with each click of the button.

ECW or JP2 Toolbar

- Add ECWP – browse an ERDAS APOLLO Server and open an image via the ECWP protocol.

Viewing Images

If you have registered the application to handle ECW and JPEG 2000 images on install, double clicking an image in the file browser will automatically launch the application. You can open an image from the local machine from within the application by clicking the "Open" button on the main toolbar or selecting File → Open from the main menu. The selected image will appear in a new map window.

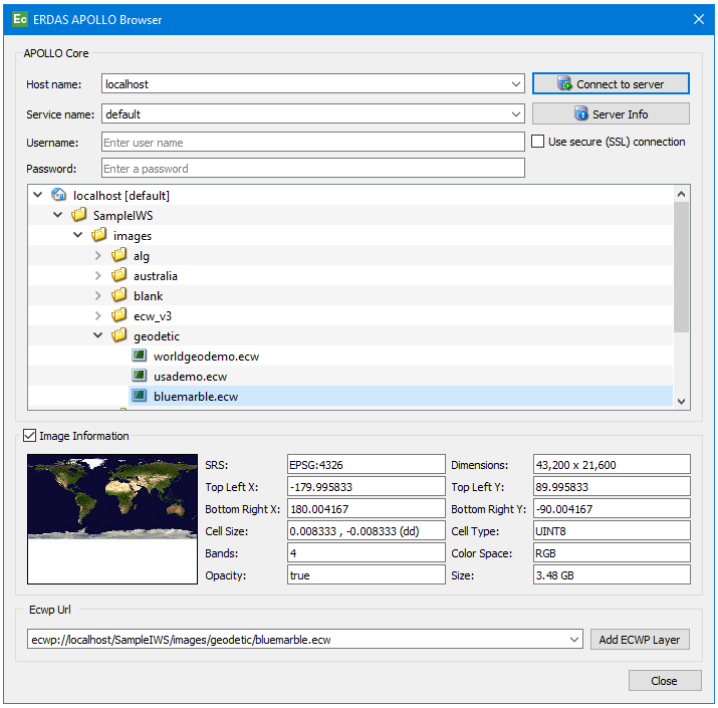
You can also drag and drop an image from the operating system file browser, onto the main application and it will open in a new window.

To add an image as a new layer in the current map view, drag and drop the image from the file browser onto the workspace window (tree view with map and layers). If no map window is currently open, a new window with the image is created. If a map window is active, the image will be added as a new layer to that map view.

Viewing ECWP Images

Select the "Add ECWP" from the main toolbar, or right click "Map" icon in the Workspace view and select "Add Layer → Add ECWP Image" from the menu. You can also select "Add Layer" from the Layer menu on the main toolbar. The ERDAS APOLLO Browser window will open.

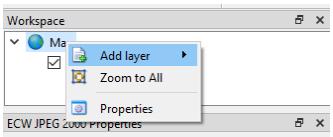
The viewer supports opening ECWP streamed images from ERDAS APOLLO 2020.

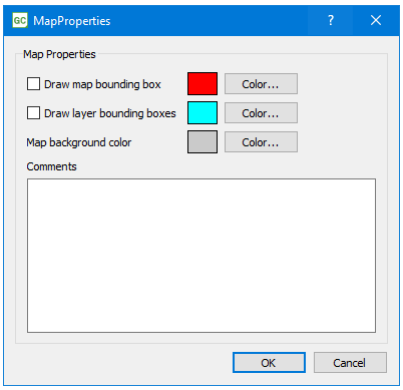


1. Enter a server and (optional) port name. for example, ~~demo-apollo.hexagongeospatial.com~~ https://demo-apollo.hexagongeospatial.com/erdas-apollo/index.html#/
2. If using SSL (ECWPS), select the “Use secure (SSL) connection”.
3. Optionally select “Server Info” to display the ERDAS APOLLO Server version and metadata.
4. Select the service name from the drop-down combo box (if not using the default service) from which you would like to browse images.
5. Open the root node and browse through the tree hierarchy to find the image you want to display.
6. Select an image to show its metadata and thumbnail. Double click an image to add it to the current map display and close the browser. Alternatively, select “Add ECWP Layer” then continue browsing and add further images as new layers to the map. When done, select Close.
7. You can also past an ECWP(S) URL directly into the URL box at the bottom of the dialog and select “Add ECWP Layer” if you know the URL of the image resource.

Map Properties

Use the map properties dialog to edit some basic map properties. To open, right click the “Map” item in the Workspace Window and select “Properties”.





Current properties settings:

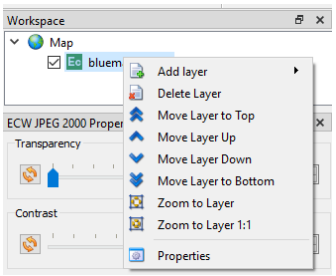
- Draw map bounding box and colour
- Draw layer bounding boxes and colour
- Comments about the map (saved into the map XML)

Layer Properties

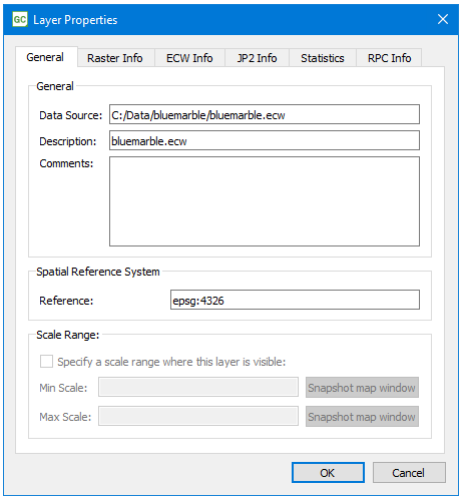
The layer properties dialog allows editing of some basic layer properties. To open, right click the layer in the Workspace Window and select “Properties”.

To open the layer properties dialog:

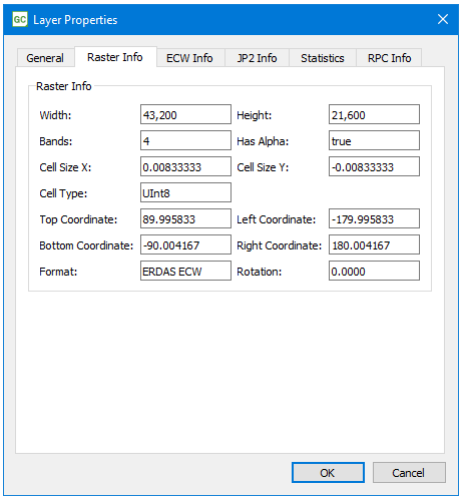
- Select the layer in the Map Workspace.
- Right click on the Layer and select Properties, or
- Go to the Layer menu on the main toolbar and select Properties.



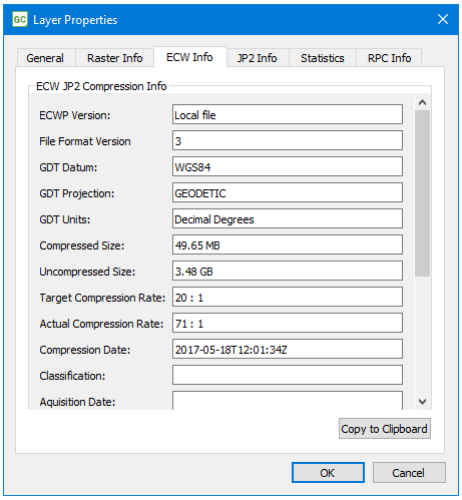
The General tab of the properties dialog shows the data source, the description and comments. It also shows the native spatial reference system for the layer.



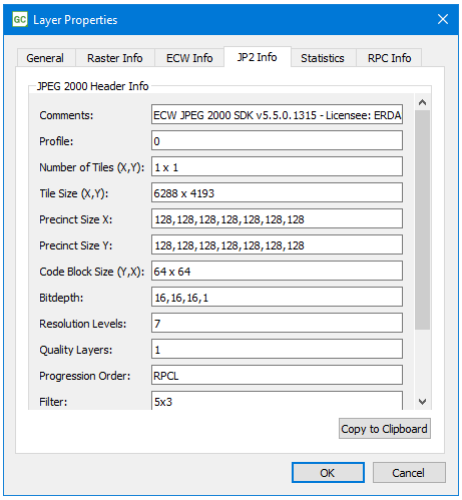
Raster Info tab displays the generic raster properties for the layer.



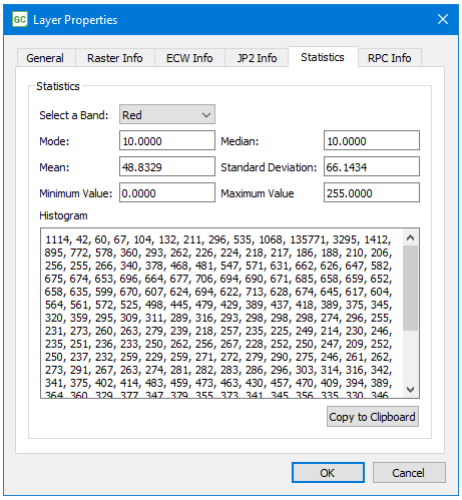
ECW Info tab displays the metadata specific to ECW files, including the ERDAS ER Mapper GDT datum and projection (if defined) as well as compression information, and ECW version 3 metadata such as acquisition date, classification etc. Select “Copy to Clipboard” to copy the information in a text format to the system clipboard.



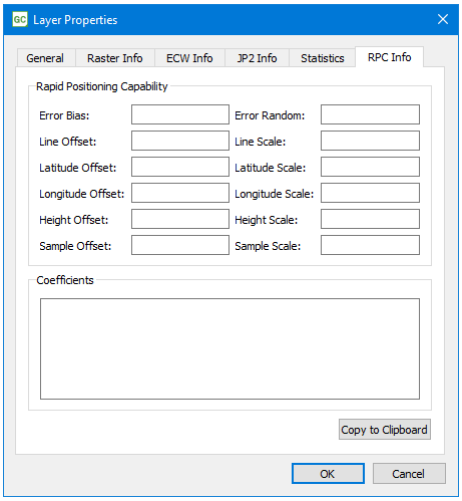
JP2 info tab displays the metadata specific to JPEG 2000 files, such as the profile, tile and precinct sizes, resolutions and quality layers etc. Select “Copy to Clipboard” to copy the information in a text format to the system clipboard.



Statistics tab displays the embedded statistics information contain in ECW v3 files (if present). You can select which band to show from the drop-down combo box containing the band descriptions. Select “Copy to Clipboard” to copy the information in a text format to the system clipboard.

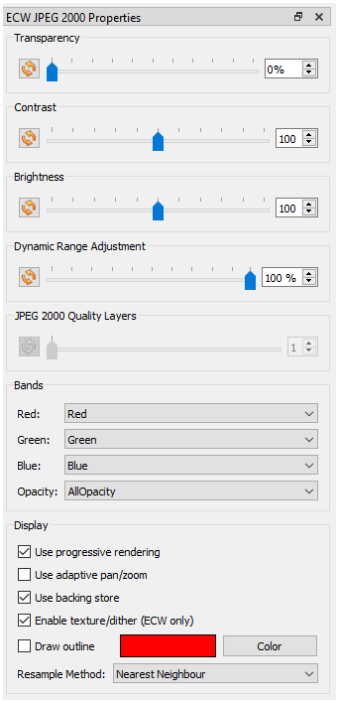


RPC Info tab displays the Rational Polynomial Coefficients and associated metadata that can be stored in ECW v3 files. Select “Copy to Clipboard” to copy the information in a text format to the system clipboard.



ECW Layer Properties

ECW JPEG 2000 Properties panel displays extra properties specific to the ECW or JP2 layer type display.

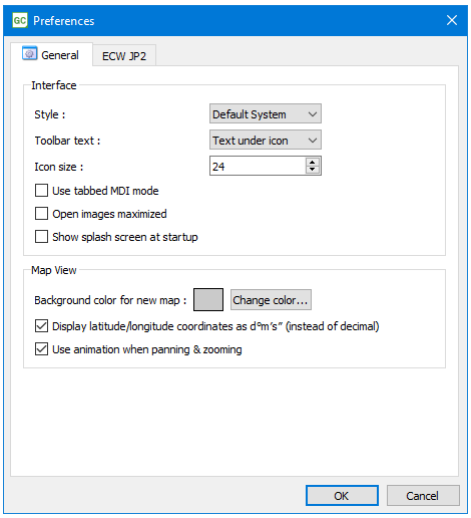


The first group of properties allow some post processing on the layer, these are transparency, contrast, brightness, dynamic range adjustment, JPEG 2000 quality layers. The second set of properties allow the user to specify band combinations and opacity (alpha) channel display. The third set of properties are custom layer properties for ECW and JP2 files. These are:

1. Use Progressive Rendering: display imagery as it comes in from the internet. Unselecting this will cause each pan and zoom to block until all imagery has been downloaded and decoded.
2. Use Adaptive pan/zoom: when in non-progressive mode, use lower resolution decoding to speed up display.
3. Use Backing Store: draw a low-resolution bitmap in the background to fill the display area
4. Enable Texture/dither (ECW only): apply texturing to the decoded image. For some images (especially over compressed images) this may provide a better-quality image display.
5. Draw outline: draw a coloured box around the extends of the layer.

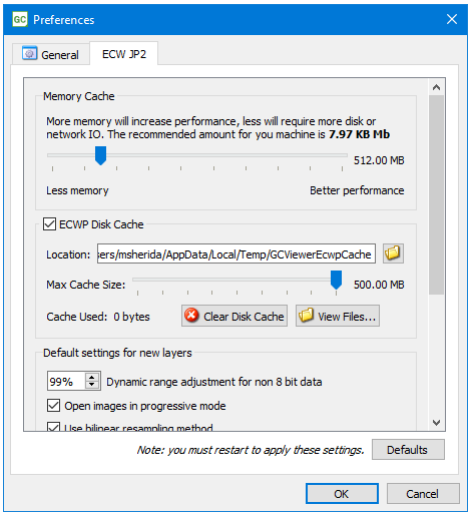
Preferences

Open the preferences dialog by selecting “File” then “Preferences” from the main menu bar.



General tab allows to set some preferences for the whole program.

- Style: select the icon style in the toolbars.
- Toolbar text: show text under or next to icons.
- Icon size: change the size of the main toolbar icons.
- Use tabbed MDI mode: image windows are aligned along the top of the main window with tabs. If unselected, traditional MDI (multiple document interface) will be enabled.
- Open images maximized: when in MDI mode, maximise the image window when creating a new map.
- Show splash screen at start-up.
- Background colour for new map: select a map background colour default.
- Display latitude/longitude coordinate as DMS (instead of decimal degrees).
- Use animation when panning and zooming: when you double click the map to zoom in, or use the keyboard shortcuts (up, down, left right), or use the mouse scroll wheel, the map will use animated panning. Unselect this to jump directly to the new map location/view.



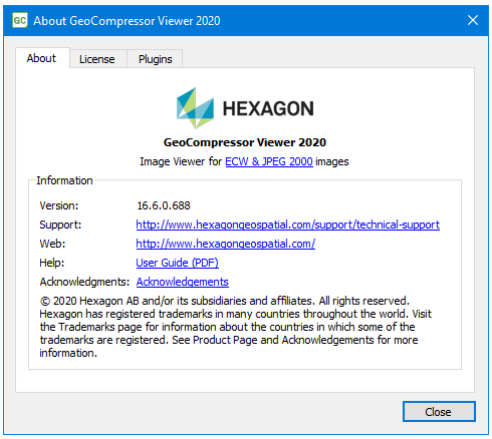
ECW JP2 tab is specific to the ECW imagery layer plugin. The preferences are:

1. Memory Cache: choose the maximum amount of system memory that will be allocated to the ECW cache.
2. ECWP Disk Cache
    a. Location: the location on the local file system to store the cache files. Default is the user temp dir.
    b. Max Cache Size: Maximum amount of disk space to use for the persistent cache.
    c. Clear Disk Cache: clear the contents of the cache to reclaim disk space.
    d. View Files: view the files in the disk cache.
3. Dynamic range adjustment for non 8-bit data: when viewing imagery with higher bit depth per channel, select a transform percentage when scaling down to 8-bit for display. This is effectively a simple DRA (dynamic range adjustment) using a percentage left-right clip.
4. Open images in progressive mode: display imagery as it comes in from the internet. Unselecting this will cause each pan and zoom to block until all imagery has been downloaded and decoded. The layer preference over-rides this default value.
5. Use bilinear resampling: smooth imagery when resampling using a bilinear filter. The layer preference over-rides this default value.
6. Use adaptive pan/zoom: decode at lower view resolutions for faster display. The layer preference over-rides this default value.
7. Enable texturing when decompressing ECW images: apply texturing to the decoded image. For some images (especially over compressed images) this may provide a better-quality image display. The layer preference over-rides this default value.
8. Force ECWP version 2 connections: fallback to the old protocol (current is v3). This is not recommended and is for testing purposes only.
9. Use WinHTTP in the ECWP Client: Use WinHTTP over WinInet. This is not recommended and is for testing purposes only.
10. ECWP HTTP User-Agent: set the user agent for ECWP streams. Some firewalls may block certain user agents. You can work around by setting a custom user agent here.
11. Log Information Level: set the level for logging. Change this only when diagnosing problems, as it impacts performance of the ECW and JPEG 2000 decoders. Default level is “Info”.

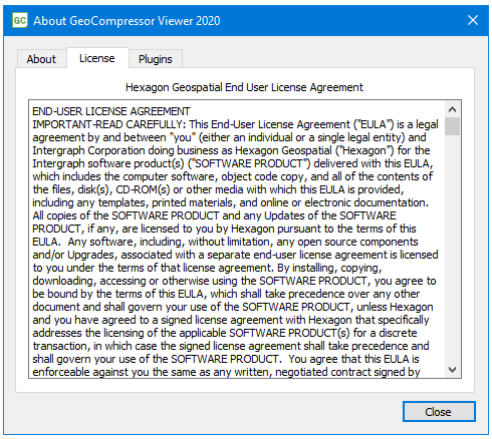
About

Select “About” from the Help menu on the main window to display the about dialog.

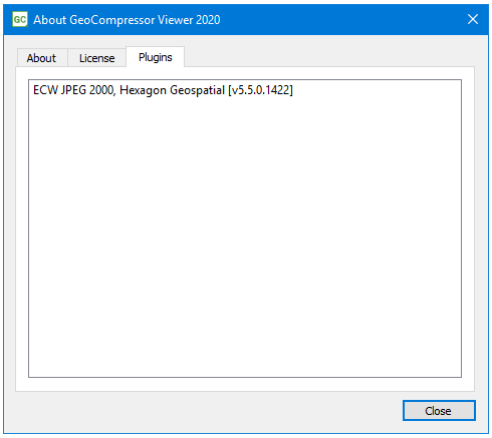
About: Version information is displayed, as well as links to the technical support web site, the Hexagon Geospatial web page, the User Guide, and the Acknowledgements can be found on this page.



License: displays the full text of the End User License Agreement.



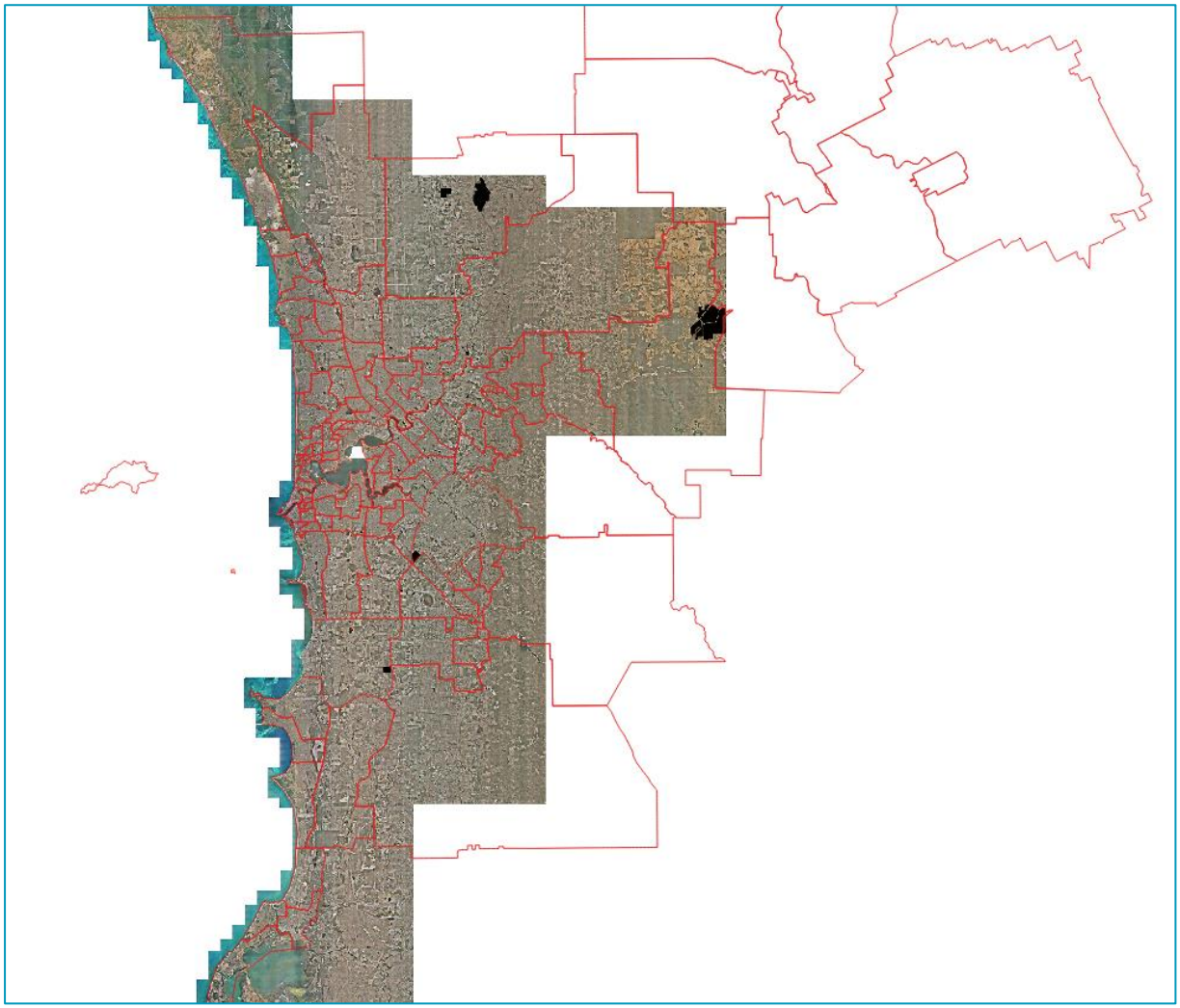
Plugins: displays version information about the installed layer plugins.



Appendix A: Mosaic to Multiple Output Workflow Example

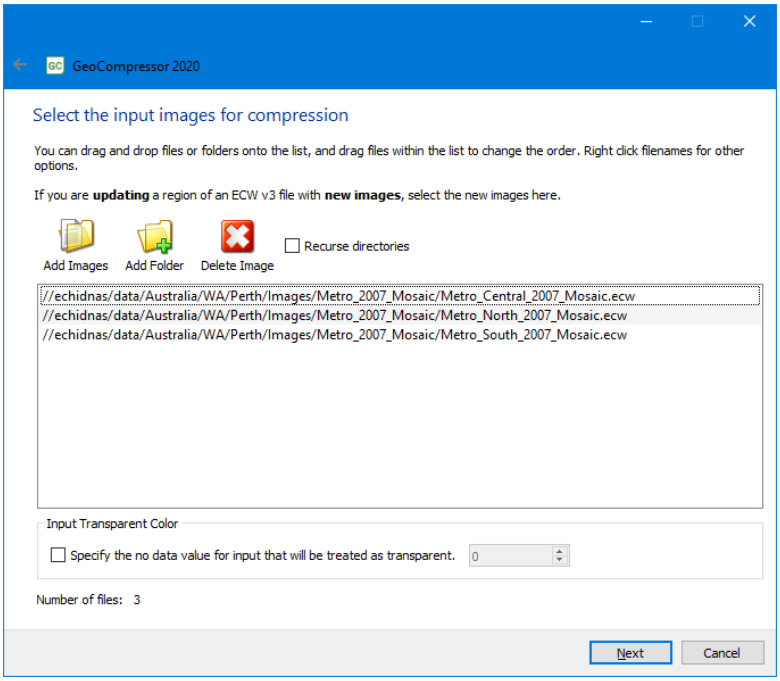
A workflow starting in GeoCompressor 2018 is the production of many output files from an input mosaic (or single image), where each output file is clipped (and opacity channel generated) to a polygon defined in a shapefile. For example, you may have 1000 TIFF files as inputs to a mosaic, and you have a shapefile containing 100 polygons, where each polygon represents a ward or county boundary. The output will be 100 ECW or JP2 files, each one clipped to the shape of the polygon.

Given the following example, the input mosaic of files will be compressed to a set of output files, where each output file corresponds to a polygon in the shapefile, in this case, a ward boundary. Outputs will only be generated where there is valid data in the intersection between the polygon and input raster dataset.

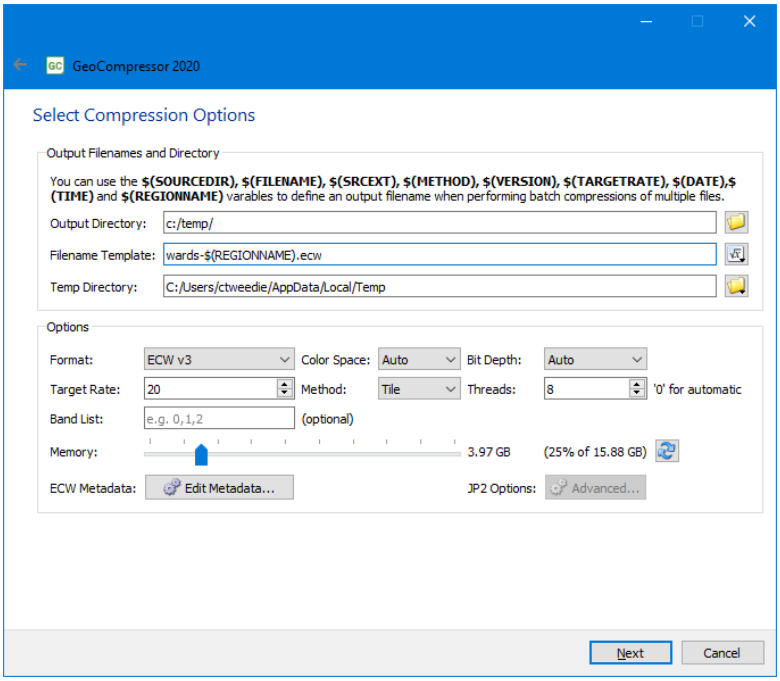


1. Open the compression wizard and select “Mosaic a set of images to a single or multiple output images”. Click next.

2. Select your inputs as normal on the inputs page. Click next.

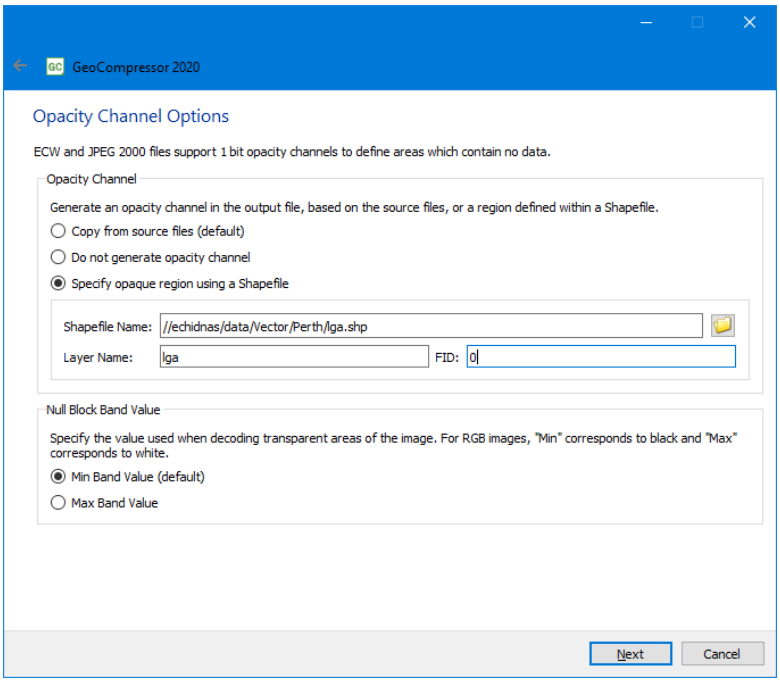


3. Select the output options, specifying the format and compression parameters. To prevent naming collisions of the multiple output files, ensure $(REGIONNAME) is specified. Click next.

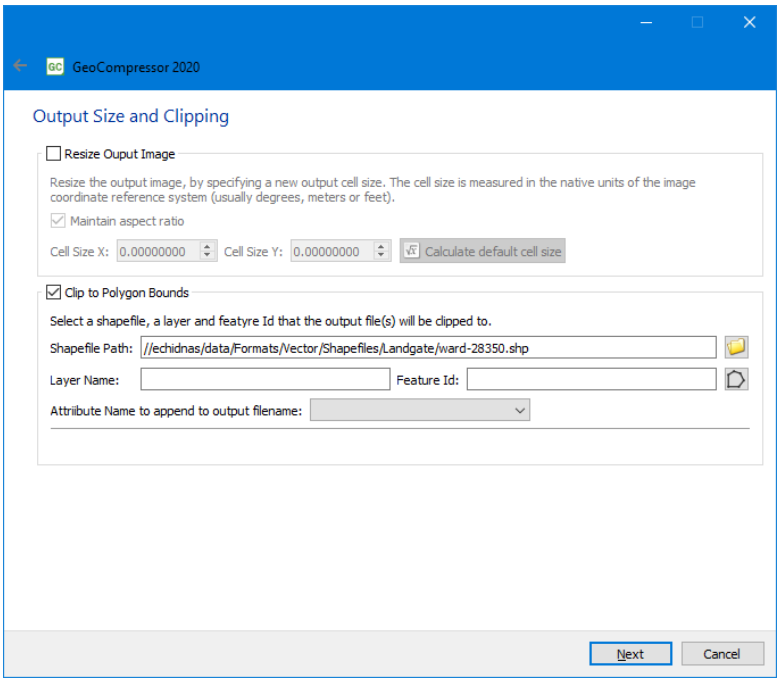


4. On the “Opacity Channel and Options” page, select “Specify opacity region using a Shapefile”, select a shapefile and enter 0 for the FID (it will be replaced with the polygon id specified in the later page).

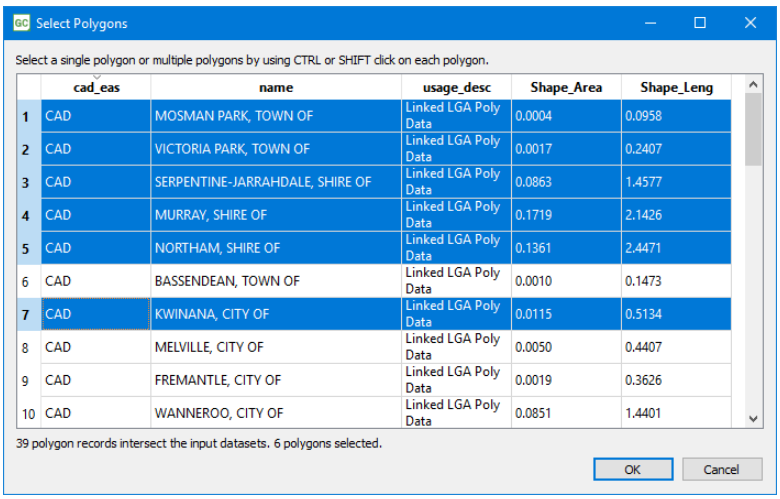
Note for this workflow, the shapefile selected here must be the same as the one selected on the next wizard page. Click Next.



5. “Output Size and Clipping” is a new page for the new workflow functionality. Select “Clip to Polygon Bounds” and select the same shapefile from the previous opacity channel options page.

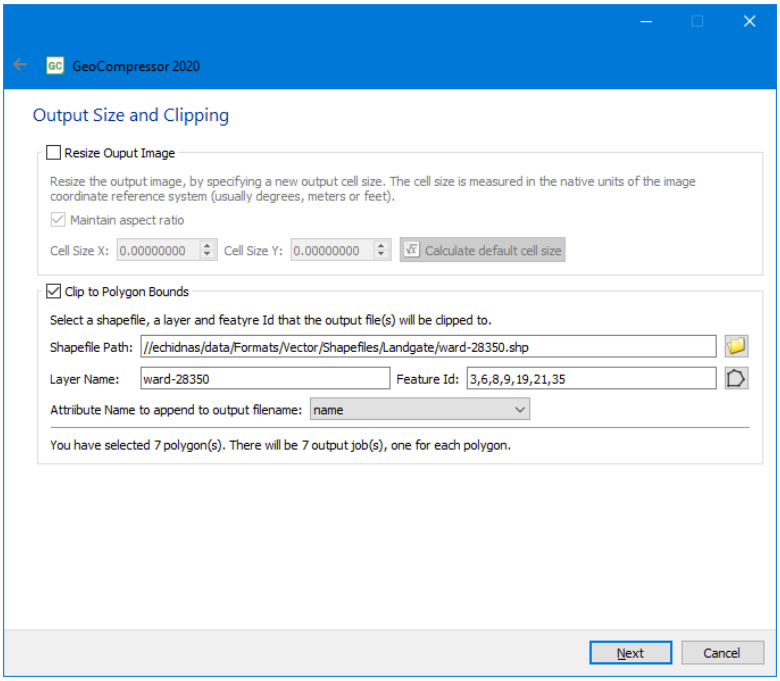


6. Click the “Select Polygons” button. A new polygon selector window opens, you can select 1, or more polygons here (using CTRL or SHIFT click). For each polygon you select, one output file will be generated. Take note of the column names, you will need to select an attribute name to use its value to generate a unique output filename. In this example, the unique attribute “name” will be used in the output filename specification. Click ok to continue.

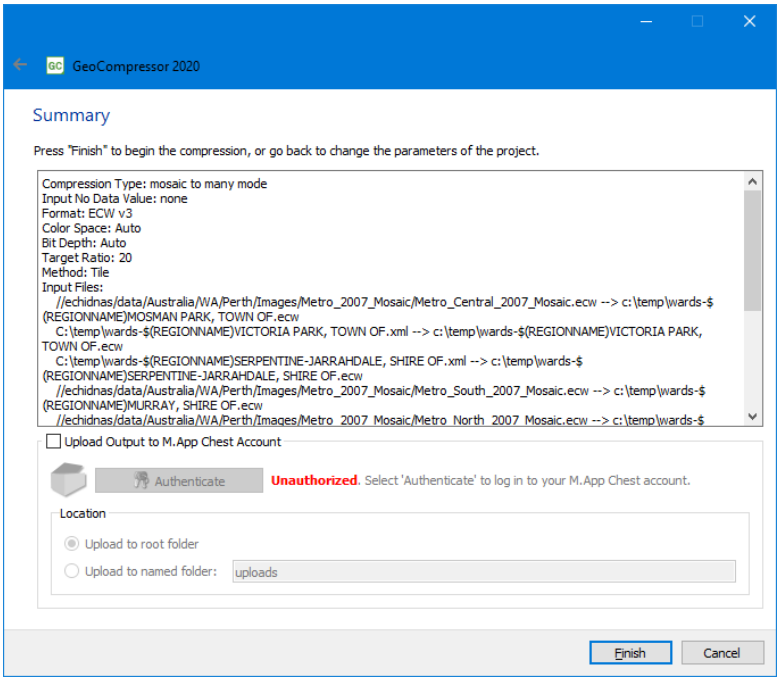


	cad_eas	name	usage_desc	Shape_Area	Shape_Leng
1	CAD	MOSMAN PARK, TOWN OF	Linked LGA Poly Data	0.0004	0.0958
2	CAD	VICTORIA PARK, TOWN OF	Linked LGA Poly Data	0.0017	0.2407
3	CAD	SERPENTINE-JARRAHDALE, SHIRE OF	Linked LGA Poly Data	0.0863	1.4577
4	CAD	MURRAY, SHIRE OF	Linked LGA Poly Data	0.1719	2.1426
5	CAD	NORTHAM, SHIRE OF	Linked LGA Poly Data	0.1361	2.4471
6	CAD	BASSENDEAN, TOWN OF	Linked LGA Poly Data	0.0010	0.1473
7	CAD	KWINANA, CITY OF	Linked LGA Poly Data	0.0115	0.5134
8	CAD	MELVILLE, CITY OF	Linked LGA Poly Data	0.0050	0.4407
9	CAD	FREMANTLE, CITY OF	Linked LGA Poly Data	0.0019	0.3626
10	CAD	WANNEROO, CITY OF	Linked LGA Poly Data	0.0851	1.4401

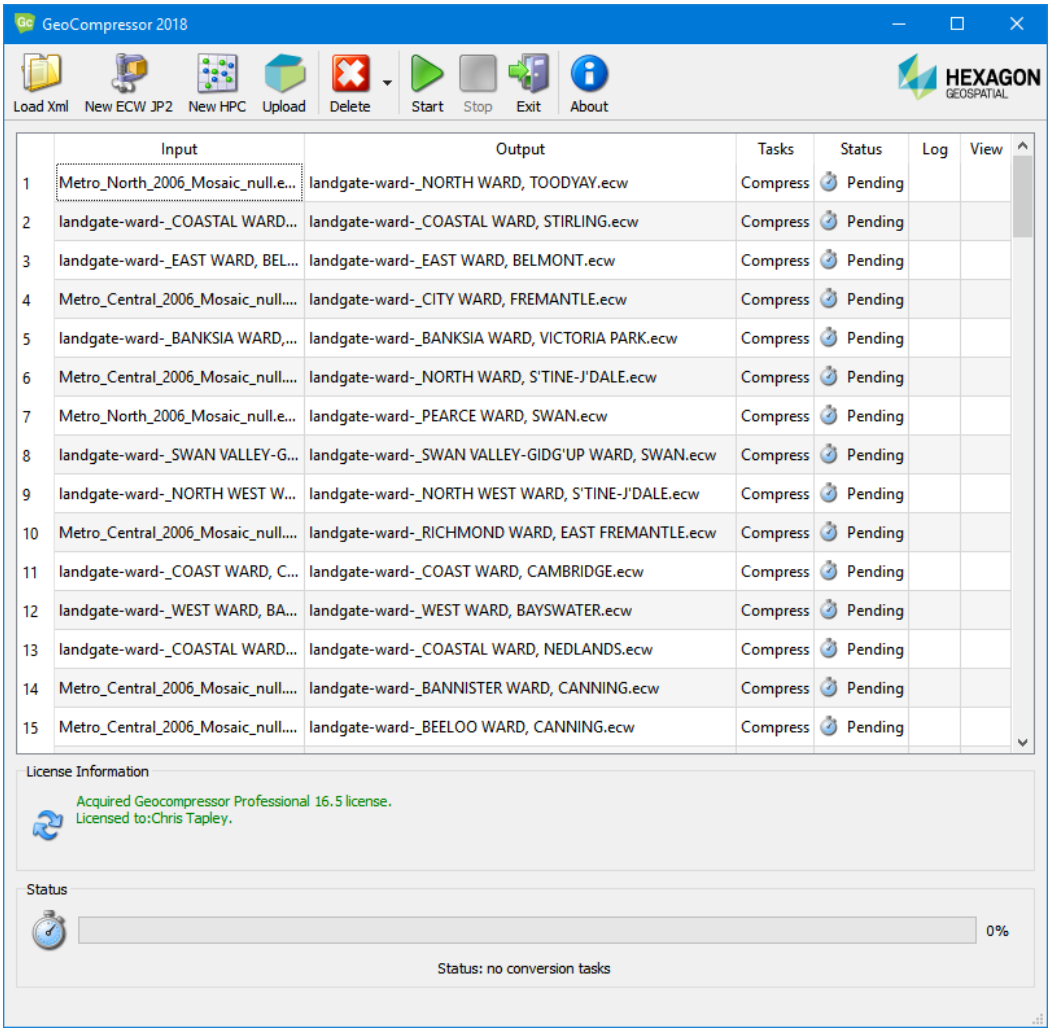
7. The Output Size and Clipping page now displays the number of polygons selected and the number of output files to be generated. Select “name” from the drop-down list of attribute names as the filename qualifier, these values will replace the $(REGIONNAME) templated filenames. Click next.



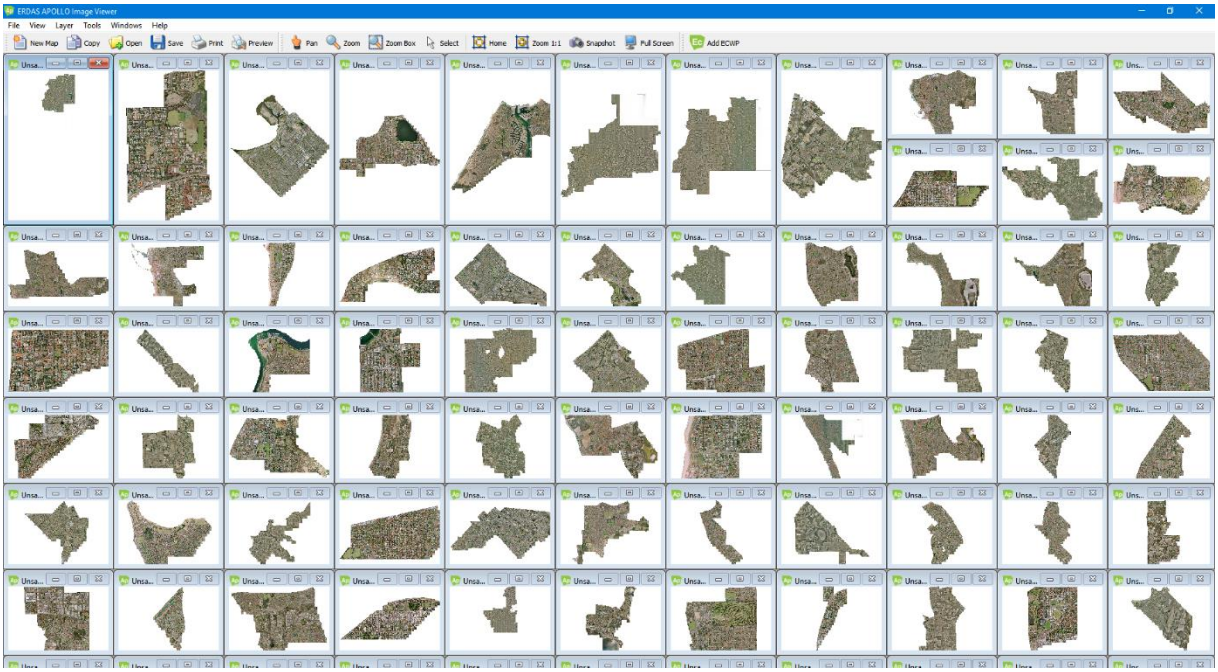
8. Select Next to move through to the summary page, then select “Finish”.



9. The task list shows the compression task queue, one task for each polygon, with the output name appended with the attribute value of each polygon. Select “Start” to begin the compression process.



10. When the compression tasks have completed, you should have an ECW or JPEG 2000 file corresponding to each polygon, similar to the following:


ERDAS APOLLO Image Viewer
File View Layer Tools Windows Help
New Map Copy Open Save Print Preview Pan Zoom Zoom Box Select Home Zoom 1:1 Snapshot Full Screen Add ECWP

Appendix B: Troubleshooting

Low Memory Issues During Compression

GeoCompressor uses Intel Thread Building Blocks for memory allocation and management, which is faster than the normal system routines under high multi-threaded load. However, it may use more memory than the normal system routines. If you are running low on memory or compressions fail with a message about not enough system memory, you can remove the Thread Building Blocks library and the compressor will run as usual, however it may be slower, but it will use less memory. The library can be found at:

<INSTALLDIR>/bin/tbbmalloc.dll (Windows)

<INSTALLDIR>/lib/tbbmalloc.so (Linux)

You should ensure you make a backup of the library before deleting it so you can restore it at a later time.

Unexpected Application Closure / Crash

In the unlikely event of a processing job suddenly exiting for no apparent reason, refer to your GeoCompressor installation log/crashdump folder. In the event of a crash, .dmp files (Windows) will be written that allows the product team to triage and understand what caused the issue. Please submit these files to Hexagon Geospatial support team if found.

Appendix C: FAQ

General

Should I use GeoCompressor or ERDAS IMAGINE Professional?

The answer is really dependant on the required functionality. If basic mosaicking or batch compression is required, GeoCompressor is perfectly suitable. If you require more control over cut-lines, colour balancing and need to perform other image/point processing tasks prior to compression, ERDAS IMAGINE is recommended. GeoCompressor is designed to complement existing workflows rather than address the end-to-end image or point cloud needs.

Are files created by GeoCompressor any different to those from ERDAS IMAGINE?

No. There may be slight variations to the metadata stored in the file headers, but otherwise an ECW, JPEG2000 compressed image or HPC Point cloud will be identical regardless what application is used. Both applications use the same underlying toolkits for conversion/compression.

What relationship does GeoCompressor have with the legacy ERDAS ER Mapper Image Compressor application?

None. GeoCompressor is a completely redesigned and rebuilt compression application. Despite the similar product naming, the capabilities are not equivalent and is not a direct upgrade from the legacy product.

Can GeoCompressor be used on a headless Linux machine?

Yes. Starting with 2020 we now deploy RPM installers that fully support headless, or command line only execution. In this deployment mode, we still deploy the user interface however does not need to be used.

What window managers are supported on Linux?

GeoCompressor GUI is built using the Qt cross-platform library and has been tested across Gnome, KDE and Unity platforms.

Why are some Linux platforms listed as viable?

Due to time constraints, platforms that are viable are deemed to be usable and in some cases tested however not to the same degree as fully supported platforms and may require additional libraries to function.

Image Compressor

What advantages does GeoCompressor provide over using the ECWJP2 SDK within GDAL?

Although GDAL has excellent support for the ECWJP2 SDK, a key feature it lacks is the multi-threaded tile encoder and region management for null block and opacity generation. There are also other performance implications that limit throughput if GDAL is used for both reading and writing. GeoCompressor will remain faster at compressing equivalent input even if the GDAL reader is used. Mosaicking is significantly faster.

What mosaic file format gives the best performance?

GeoCompressor supports both ERDAS ER Mapper ALG and GDAL VRT file formats. Both support a wide range of dynamic capabilities including the ability to mosaic separate files together to present them as a single virtual file or mosaic. The Image Compressor mosaicking process uses the ERDAS ER Mapper ALG engine internally for this capability and was chosen for performance reasons and because its thread-safe. VRT is supported however is known to perform slower than an equivalent ALG especially for large input sizes. VRT files can also only be compressed using the line encoder.

Why can't I define a multi-polygon for opacity/null definition?

You can! This previous limitation has been resolved in the 2015 release. You can now provide vector regions with multi-part shapes (e.g. holes or multiple geometries).

How does GeoCompressor calculate pixel/polygon intersections?

GeoCompressor calculates the intersection based on the top left coordinate of each pixel. Therefore you may need to adjust your polygons by half a pixel or more to ensure that the bottom right corner encapsulates the entire raster data you want to include.

I use complex ERDAS ER Mapper Algorithms that link to Virtual ERS rasters, kernels and so on, will GeoCompressor compress these files?

Yes. GeoCompressor uses the full ERDAS ER Mapper Library however in a 64-bit environment. GeoCompressor can be used to compress large datasets where memory requirements exceed 32-bit limits but still benefit from the powerful visualization tools that ERDAS IMAGINE provides.

Is GeoCompressor faster than Product X?

GeoCompressor compresses faster than the legacy ERDAS ER Mapper 7.2 (64-bit), ERDAS ER Mapper Mosaic Balance Compressor (MBC), Safe FME, and Global Mapper products, all of which are used heavily throughout the industry for compressing to ECW and JPEG2000. These products use the legacy ECWJP2 SDK v3.x, released over 13 years ago (2006). In comparison, ERDAS IMAGINE and GeoCompressor are always built on the latest ECWJP2 SDK and benefits from years of further research, optimizations for latest hardware, bug fixes and other improvements. GeoCompressor is maintained by the same development team as the ECWJP2 SDK ensuring the product is the reference, best-of-breed implementation of the underlying compression engine.

Why does GeoCompressor prevent some input formats from being compressed in tile mode?

Using the tile compression method forces multi-threaded accesses on the input data. For a variety of reasons many input formats are not thread-safe. Where we are aware of formats that are not thread-safe we default to the line compression method and will return an error if tile is used.

The GDAL VRT format is one such example of an input format that compresses without issue with a single thread however causes significant problems when called from multiple threads. This problem is a general GDAL Data Reader problem and is not specific to GeoCompressor, see GDAL documentation. As improvements are made to the Data Readers like GDAL, GeoCompressor will re-enable these formats.

Why is georeferencing sometimes lost in the output files?

GeoCompressor relies on the Data Readers to provide georeferencing information in order to resolve back to an EPSG code. For a variety of reasons this process is not always successful in which case GeoCompressor will create output files in a “WGS84/LOCAL”. This definition can be altered to the correct EPSG code by updating the file header after the fact; a recompression is not required. Or alternatively the –srs command line parameter can be defined prior to compression to force the output definition to the given EPSG code.

If no georeferencing is detected at all and the input data is raw, “RAW/RAW” projection/datum pairing will be written to the output.

To confirm, check the Compressor log and look for the “Projection” value set within the “Input Data” metadata area.

What settings should I use to get best throughput?

This question is dependent on too many variables and can only be answered after benchmarks and comparisons conducted on customer hardware with customer input data. Contact Hexagon Geospatial Support for assistance and include an example logfile of a compression task you would like to optimize.

Are pyramids or overviews required for compression?

No. GeoCompressor only reads the input data at native resolution. If files such as RRD, OVR or internally defined overviews are present they will be ignored. Generating these files is a redundant processing step for compression workflows; they do not speed up the compression of ECW or JPEG2000 files so is not recommended unless required to QA prior to compression. It is a waste of time and storage space.

Why is the output actual compression ratio significantly different to the target when using ECW v3 null blocks?

Depending on the region supplied and the amount of area defined as null, it is not uncommon to specify 20:1 target and obtain an actual compression rate of 50:1 or more. When null blocks are enabled the amount of data written to the output file can be substantially reduced, which is reflected in the actual compression ratio since this divides the uncompressed input size by the output size. It makes no allowances between null or data areas.

This behaviour can be quantified by customers using the ERDAS APOLLO Image Quality utility or by your own visual quality assessment by overlaying a sample image with and without null blocks enabled. The image quality of the “data” areas will remain identical to the same image compressed to 20:1 without null blocks even though its actual target rate may be closer to the target rate and the file size being substantially different.

When should null blocks be enabled?

Null blocks were designed to shrink the output file size and speed up compression speed, but these benefits are directly related to the characteristics of the input. To help identify when null blocks should be enabled, the compression output will report the “Ratio to data” as well as the number of vertices that make up the selected region. Where the ratio to data is low enabling null blocks will not yield much change to the output file size and in many cases will increase compression time due to the additional spatial intersection checking. Similarly if the number of vertices is in the thousands, performing spatial intersection tests against very complex polygons could adversely impact the compression time negating the compression speed improvements.

The degree to which these trade-offs affect throughput is heavily dependent on the input data but also the hardware that the compression is performed on. The only reliable way to answer this question is to perform two identical compression tests, one with –opacity 0 (or with no region specified) and one with –opacity 2. This will allow direct comparison of the benefits that null blocks provide. See the Null Block Analysis chapter for more information and comparisons.

Why does the ImageCompressor need to calculate min/max values?

To compress signed or unsigned 16-bit output the compressor needs to know the data range to ensure high quality output when the actual data range is a smaller sample, for example 11-bit stored in a 16-bit range. This step is not required for 8-bit output.

Depending on the size and format type this range calculation may take a few minutes before the actual compression begins. This calculation uses an approximate algorithm by taking an overview image of approximately 2,500 pixel samples to determine the range. If the range reported appears wildly inaccurate it is recommended to calculate full statistics prior to compression and GeoCompressor will read existing statistics when detected.

GeoCompressor supports reading statistics stored in *.aux.xml, *.aux, ECW v3, ERS and ALG files.

When compressing large inputs why is there a delay when the process reaches 95%? Or the opposite problem where the progress completes 95-100% extremely quickly?

For large inputs 100gigapixels or more there can be a noticeable delay when reassembling the output file. We now report the “Re-assembly time” in the output report and count this phase as part of the compression time. Previously re-assembly would be performed once progress was at 100%, which created unnecessary confusion with customers who believed the process had hung. If the re-assembly time is greater than a few % of the total duration, consider improving the disk storage sub-system as the bottleneck clearly is reading from the temporary location and writing to the output file location.

Why do I see a warning regarding NUMA?

GeoCompressor detects when multiple NUMA nodes are enabled and defaults to the thread count of 1 NUMA node only. This is done for scalability reasons as the current tile encoder cannot scale across processor groups. Reducing the –threads to the single processor core count will remove this warning.

Why do I sometimes see background pixels along edges of the opacity region with files that contain an opacity channel?

In the image below, the background of the map is red (for demonstration purposes), but the background colour of the ECW image is defined as white at compression time. The polygon defining the region to use as an opacity channel, lines up exactly with the edge of the raster data, causing white to bleed into the image.

This is due to the nature of the wavelet transform, the colour (white) on the background edge of the image will affect the pixels along the “image” edge when the discrete wavelet transform is done on compression. This often happens when the image data bounds and the opacity region are exactly aligned to the same pixel.

To alleviate this affect, make the opacity channel defining region slightly smaller than the raster region, leaving a few pixels as a boundary is usually enough to eliminate the effect and get a seamless DWT (discreet wavelet transform). If the polygon defining the region is always smaller than the input raster area, you will not see this problem.

Point Compressor

What is the Hexagon Point Cloud (HPC) format?

HPC is a point cloud format that is based on licensed technology and used across a variety of Hexagon software products. The optimized format contains internal levels of detail (LOD) and due to its patented storage and rendering engine is able to be streamed in a client to server environment (currently ERDAS APOLLO). With respect to ERDAS APOLLO and ECW, HPC provides a rapid and effective manner to disseminate massive point clouds over ECWP, a protocol previously used to stream imagery, or via 2D ortho views using OGC WMS or WMTS protocols.

Why does specifying a smaller point resolution increase the output HPC size if the point count is the same?

HPC uses voxels as part of the compression algorithm so if a smaller value is specified than the actual point density of the input, additional (redundant) voxels must be created in the output that increases the size even though the total number of points remains unchanged. Although PointCompressor uses the reported minimum input scale factor as a guide, it’s common for many applications writing LAS to use a smaller value than the true density of the points. It is recommended to adjust values until the output point count no longer matches the input point count if creating the smallest HPC file possible is the goal.

Appendix D: ECW Header Editor CLI Parameters

The ECW header editor allows you to edit the georeferencing information contained in an ECW or JP2 file header without recompressing the file.

To open the Help for the generic parameters for ECW or JP2 files use the following command line:

```
> ECWHeaderEditorCLI.exe --header –help


Header editor operation:
  --help                                Display header editor help message
  --input arg                           Path of the dataset to update
  --projection arg                      The new projection string
  --datum arg                           The new datum string
  --origin-x arg                        The new X coordinate of the top-left
                                        corner of the top-left pixel
  --origin-y arg                        The new Y coordinate of the top-left
                                        corner of the top-left pixel
  --cell-increment-x arg                The new world units per pixel in the X
                                        axis
  --cell-increment-y arg                The new world units per pixel in the Y
                                        axis
  --update-ers arg (=0)                 Should a .ers side-car file also be
                                        updated
  --allow-invalid-datum-projection-pairs arg (=1)
                                        Allow any datum and projection
                                        combinations in the dataset
```

To open the Help for the specific parameters for JP2 files use the following command line:

```
> ECWHeaderEditorCLI.exe --jp2 --help

JP2 advanced operations:
  --help                              Display JP2 advanced operations help
                                      message
  --input arg                         Path of the JP2 dataset to update
  --show-cdef-box                     Show the cdef box contents
  --validate-cdef-box                 Validate that the cdef box contents are
                                      correct
  --correct-cdef-box-rgb-in-multiband Correct the cdef box where RGB channels
                                      are associated to bands for non RGB
                                      datasets
  --backup-dataset arg (=1)           For operations that modify the input
                                      backup the file first
```

Support

GeoCompressor product support is available to all customers with an active GeoCompressor subscription or Software Maintenance on ERDAS APOLLO. See Hexagon Geospatial Support page for more information on how to raise support requests.

When reporting problems, please include the GeoCompressor output log, which will include all relevant information to help diagnose possible causes. Input data may be required to reproduce.

About Hexagon

Hexagon is a global leader in sensor, software, and autonomous solutions. We are putting data to work to boost efficiency, productivity, and quality across industrial, manufacturing, infrastructure, safety, and mobility applications.

Our technologies are shaping urban and production ecosystems to become increasingly connected and autonomous — ensuring a scalable, sustainable future.

Hexagon's Geospatial division creates solutions that visualize location intelligence, delivering a 5D smart digital reality with insight into what was, what is, what could be, what should be, and ultimately, what will be.

Hexagon (Nasdaq Stockholm: HEXA B) has approximately 21,000 employees in 50 countries and net sales of approximately 4.4bn USD. Learn more at hexagon.com and follow us @HexagonAB.