## OptimizeRasters

(User Documentation)

1. **Introduction**
2. [**GDAL Path**](#_[GDAL_Path])
3. [**Override the default GDAL path**](#_Override_the_default)
4. [**Setup S3 Credentials**](#_[Setup_S3_credentials])
5. [**AWS *standards to manage S3 credentials***](#_1.AWS_standards_to)
6. [**Override AWS credentials**](#_2.Override_AWS_credentials.)
7. [**AWS credentials usage**](#_3._AWS_credentials)

**Introduction**

Optimize raster is a command line tool that converts raster from one format to another, the output format that is created is an optimized such that it improves the performance of the raster when they are used. It also has the option to builds pyramids on the output raster and compress the imagery so save the storage space. This tool also allows to read the data from amazon s3 bucket or upload the converted data to amazon s3 bucket. There are many parameters that user can configure based on their how they want the output raster to be written out with what compression and pyramids. It also has the capability of write the intermediate data on fast disk during the conversion process if the input is from a slower disk.

This tool accepts following command line arguments, at command prompt you can get the help as below

**<path\_to\_python.exe> <path\_to\_optimizerastes.py> --help**

-input=Input raster files directory path

-output =Output directory path

-mode =Processing mode/output format, value should be mrf, cachingmrf, clonemrf, tif. Refer below to the meaning of modes

-cache =cache output directory path

-config =Configuration file with default settings, if not defined it will pick it up from the location where the py file resides **( Optimizerasters.xml is the configuration file)**

-quality =JPEG quality if compression is jpeg

-prec=LERC precision

-pyramids =Generate pyramids- value should be true/flase

-s3input =is the input from s3 folder value can be true/flase

-s3output =is output to s3 folder the value can be true/false

-subs =Include sub-directories - value should be true/flase

-tempinput=Path to copy -input raters before conversion, if inuput is s3 this is required parameter

-tempoutput=Path to copy converted rasters before moving to output, if output is s3 it’s a required parameter

All the above arguments can be defined in the configuration file and very few can be used in the command line based on the user requirements, the sample configuration file has all the value with comments explaining what each of them means

Here are some of the commonly used commands to convert the data

1. **As a user I would like to convert input tif file to MRF with building the pyramids**

Things to set in the config file,

* Set the **RasterFormatFilter** to tif (note this is case sensitive so to be on safer side given in both the cases) you need to just mention the extension. Eg tif, tiff, jp2, jpeg, refer to the optimize raster.xml provided
* Some of the files should not be copied over as they are not needed like ovr, aux.xml, tfw so in the config file **ExcludeFilter** you can specify the filers that want be copied over to the output directory.
* If you want to build Pyramids in that case the **BuildPyramids** node value should be set as true
* Other parameter in the configuration files if need can be changes else the default values will be used

The command line that should be used is as below

<path to python.exe> <path to optimizerasters.py> -input=<path to input folder> -output=<path to outputfolder> -mode=mrf

1. In the above case the output is an s3 bucket following additional changes should be made to the config file ( following changes are compulsory to be updated in the config file)

* The s3 bucket name should be specified in the node **Out\_s3\_Bucket**
* The access ID which wil be used to make a connection needs to be specified in the node **Out\_S3\_ID**
* This also needs the scretkey to be specified in the node **Out\_S3\_Secret**
* In the config file you need to specify the output is s3to true in the node **Out\_S3\_Upload**
* This process creates an intermediate data so you should specify to delete that data by setting the node value to true in the node **Out\_S3\_DeleteAfterUpload**
* The s3 folder where the data needs to be uploaded **Out\_S3\_ParentFolder** ie. You need to exclude the bucket name if folder url is http://mydata.s3.amazonaws.com/abc/pqr/t

The bucket name is mydata

And the outputfolder path is abc/pqr/t

The command line that should be used is as below

<path to python.exe> <path to optimizerasters.py> -input=<path to input folder> -output=<path to s3 outputfolder> -tempoutput=<path\_to-a\_folder\_on\_localdisk> -s3output=true –mode=mrf

1. If the user has data in S3 in MRF format in that case we recommend to use clone MRF mode

In the ‘CloneMRF’ mode, a copy of the input directory is created including all auxiliary files, but excluding MRF data and index files. The MRF files are then modified to include appropriate links back to the original index and data files and appropriate cache files locations are defined.

Following changes are needed to be done

* The s3 bucket name should be specified in the node **In\_s3\_Bucket**
* The access ID which will be used to make a connection needs to be specified in the node **In\_S3\_ID**
* This also needs the scretkey to be specified in the node **In\_S3\_Secret**
* In the config file you need to specify the output is s3to true in the node **In\_S3\_Upload**
* The s3 folder where the data needs to be downloaded In**\_S3\_ParentFolder** ie. You need to exclude the bucket name if folder url is http://mydata.s3.amazonaws.com/abc/pqr/t

The bucket name is mydata

And the inputfolder path is abc/pqr/t

The command line that should be used is as below

<path to python.exe> <path to optimizerasters.py> -input=<path to s3 input folder> -output=<path to outputfolder> -tempoutput=<path\_to-a\_folder\_on\_localdisk> -s3input=true –mode=clonemrf

1. If the user has data in s3 bucket as tif file or another raster format ( eg jpeg, jp200 ) to use that data in arcgis framework we recommend to create a caching MRF. In the ‘CachingMRF’ mode, a copy of the input directory is created excluding all raster files such as TIF and JP2. These are substituted with CachingMRF files. If users do not want to replicate the data, but simply speed up repeat access then the CachingMRF mode can be used. This will create CachingMRF files that point back to the source data and appropriate cache files are defined. This has the advantage of not duplicating data but providing faster access and having requests cached to local machines. Prior to using CachingMRF pyramids should exist on the source data.

Following changes are needed to be done

* The s3 bucket name should be specified in the node **In\_s3\_Bucket**
* The access ID which will be used to make a connection needs to be specified in the node **In\_S3\_ID**
* This also needs the scretkey to be specified in the node **In\_S3\_Secret**
* In the config file you need to specify the output is s3to true in the node **In\_S3\_Upload**
* The s3 folder where the data needs to be downloaded In**\_S3\_ParentFolder** ie. You need to exclude the bucket name if folder url is http://mydata.s3.amazonaws.com/abc/pqr/t

The bucket name is mydata

And the inputfolder path is abc/pqr/t

The command line that should be used is as below

<path to python.exe> <path to optimizerasters.py> -input=<path to s3 input folder> -output=<path to outputfolder> -tempinput=<path\_to-a\_folder\_on\_localdisk> -s3input=true –mode=cachingmrf

1. There are cases when you data reside on your internal network storage, in such cases there is an network as well as storage bottleneck that affects the performance of your data, one way to reduce that is to use Split MRF. In the ‘SplitMRF’ mode, a copy of the input directory is created, but it excludes the MRF data files and instead adds links in the MRF files to point back to them.

<path to python.exe> <path to optimizerasters.py> -input=<path to s3 input folder> -output=<path to outputfolder> -mode=splitmrf

## [GDAL Path]

OptimizeRasters is distributed with the necessary GDAL binaries for its operation. The default path for GDAL binaries are located at (tools/bin) folder relative to the OptimizeRasters package root.

# *Override the default GDAL path*

To set a new path to GDAL binaries use the following key in the OptimizeRasters config file. <**GDALPATH**>XXX\_YOUR\_GDAL\_PATH\_XXX</**GDALPATH**>  
   
 If the node <**GDALPATH**> is omitted or its value is empty, OptimizeRasters will default to the package root location (tools/bin).

## [Setup S3 credentials]

# 1.AWS standards to manage S3 credentials.

OptimizeRasters supports the AWS standards to manage credentials. This means credentials can use the default environment variables (**AWS\_ACCESS\_KEY\_ID**, **AWS\_ACCESS\_KEY\_ID**) or by using the default AWS credential file located at **%USERPROFILE%\.aws\credentials**.  
Please note credentials a text/INI file without the extension.

One primary advantage of using AWS credential file to store S3 keys is for the default user access security offered by the OS. The credentials file will only be accessible to the user who already has the read/write access to the profile location. This will make sure when copying the OptimizeRasters package folder onto another machine, one does not inadvertently give out the credentials found in the OptimizeRasters config file.

# 2.Override AWS credentials.

To The bypass the default AWS standard to manage credentials, the OptimizeRasters config file can be edited to include the necessary information as shown below to have the credentials added to S3 input and S3 output storage respectively.

<**In\_S3\_Secret**>\_IN\_S3\_SECRET\_KEY\_</**In\_S3\_Secret**>  
<**In\_S3\_Bucket**> \_IN\_S3\_BUCKET\_NAME\_</**In\_S3\_Bucket**>

<**Out\_S3\_ID**>\_OUT\_S3\_SECRET\_KEY\_</**Out\_S3\_ID**>  
<**Out\_S3\_Secret**>\_OUT\_S3\_BUCKET\_NAME\_</**Out\_S3\_Secret**>

*The keys in the OptimizeRasters take the precedence over the AWS standard credential manager.*

# 3. AWS credentials usage.

If using the AWS credential manager for S3 bucket authentication, the entries related to AWS profiles need to be updated in the Optimizeasters parameter file to reference the matching profile names in the the AWS credentials file.   
  
For e.g. In the OptimizeRasters config file you can have the AWS manager specific entries updated to match the AWS profile names,

<**In\_S3\_AWS\_ProfileName**>*OptimizeRaster\_S3In*</**In\_S3\_AWS\_ProfileName**>  
<**Out\_S3\_AWS\_ProfileName**>*OptimizeRaster\_S3Out*</**Out\_S3\_AWS\_ProfileName**>

For the above entries to work, AWS profile file (credentials) has to be updated to reflect the profile names in the parameter file as shown below.

[***OptimizeRaster\_S3In***]  
aws\_access\_key\_id=XXX\_YOUR\_ACCESS\_KEY\_ID\_XXX  
aws\_secret\_access\_key = XXX\_SECRET\_ACCESS\_KEY\_XXX

[***OptimizeRaster\_S3Out***]  
aws\_access\_key\_id = XXX\_YOUR\_ACCESS\_KEY\_ID\_XXX  
aws\_secret\_access\_key = XXX\_SECRET\_ACCESS\_KEY\_XXX