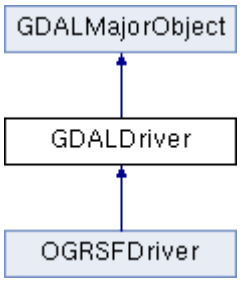
GDALDriver Class Reference

Format specific driver. More...

`#include <gdal_priv.h>`

Inheritance diagram for GDALDriver:



Public Member Functions

CPLErr	**SetMetadataItem** (const char *pszName, const char *pszValue, const char *pszDomain="") override Set single metadata item. More...
GDALDataset *	**Create** (const char *pszName, int nXSize, int nYSize, int nBands, **GDALDataType** eType, char **papszOptions) **CPL_WARN_UNUSED_RESULT** Create a new dataset with this driver. More...
CPLErr	**Delete** (const char *pszName) Delete named dataset. More...
CPLErr	**Rename** (const char *pszNewName, const char *pszOldName) Rename a dataset. More...
CPLErr	**CopyFiles** (const char *pszNewName, const char *pszOldName) Copy the files of a dataset. More...
GDALDataset *	**CreateCopy** (const char *, **GDALDataset** *, int, char **, GDALProgressFunc pfnProgress, void *pProgressData) **CPL_WARN_UNUSED_RESULT** Create a copy of a dataset. More...

▸ **Public Member Functions inherited from GDALMajorObject**

Static Public Member Functions

static **CPLErr**	**QuietDelete** (const char *pszName) Delete dataset if found. More...
static **GDALDriverH**	**ToHandle** (**GDALDriver** *poDriver) Convert a GDALDriver* to a GDALDriverH. More...
static **GDALDriver** *	**FromHandle** (**GDALDriverH** hDriver) Convert a GDALDriverH to a GDALDriver*. More...

▸ **Static Public Member Functions inherited from GDALMajorObject**

Additional Inherited Members

Protected Member Functions inherited from GDALMajorObject

Detailed Description

Format specific driver.

An instance of this class is created for each supported format, and manages information about the format.

This roughly corresponds to a file format, though some drivers may be gateways to many formats through a secondary multi-library.

Member Function Documentation

CPLErr GDALDriver::CopyFiles (const char * pszNewName,
const char * pszOldName
)

Copy the files of a dataset.

Copy all the files associated with a dataset.

Equivalent of the C function GDALCopyDatasetFiles().

Parameters

pszNewName new name for the dataset.

pszOldName old name for the dataset.

Returns

CE_None on success, or CE_Failure if the operation fails.

```
GDALDataset * GDALDriver::Create ( const char *      pszFilename,
                                   int               nXSize,
                                   int               nYSize,
                                   int               nBands,
                                   GDALDataType      eType,
                                   char **           papszOptions
                                 )
```

Create a new dataset with this driver.

What argument values are legal for particular drivers is driver specific, and there is no way to query in advance to establish legal values.

That function will try to validate the creation option list passed to the driver with the **GDALValidateCreationOptions()** method. This check can be disabled by defining the configuration option GDAL_VALIDATE_CREATION_OPTIONS=NO.

After you have finished working with the returned dataset, it is **required** to close it with **GDALClose()**. This does not only close the file handle, but also ensures that all the data and metadata has been written to the dataset (**GDALFlushCache()** is not sufficient for that purpose).

In some situations, the new dataset can be created in another process through the **GDAL API Proxy** mechanism.

In GDAL 2, the arguments nXSize, nYSize and nBands can be passed to 0 when creating a vector-only dataset for a compatible driver.

Equivalent of the C function **GDALCreate()**.

Parameters

pszFilename	the name of the dataset to create. UTF-8 encoded.
nXSize	width of created raster in pixels.
nYSize	height of created raster in pixels.
nBands	number of bands.
eType	type of raster.
papszOptions	list of driver specific control parameters. The APPEND_SUBDATASET=YES option can be specified to avoid prior destruction of existing dataset.

Returns

NULL on failure, or a new **GDALDataset**.

```
GDALDataset * GDALDriver::CreateCopy ( const char *        pszFilename,
                                       GDALDataset *       poSrcDS,
                                       int                 bStrict,
                                       char **             papszOptions,
                                       GDALProgressFunc    pfnProgress,
                                       void *              pProgressData
                                     )
```

Create a copy of a dataset.

This method will attempt to create a copy of a raster dataset with the indicated filename, and in this drivers format. Band number, size, type, projection, geotransform and so forth are all to be copied from the provided template dataset.

Note that many sequential write once formats (such as JPEG and PNG) don't implement the **Create()** method but do implement this **CreateCopy()** method. If the driver doesn't implement **CreateCopy()**, but does implement **Create()** then the default **CreateCopy()** mechanism built on calling **Create()** will be used. So to test if **CreateCopy()** is available, you can test if GDAL_DCAP_CREATECOPY or GDAL_DCAP_CREATE is set in the GDAL metadata.

It is intended that **CreateCopy()** will often be used with a source dataset which is a virtual dataset allowing configuration of band types, and other information without actually duplicating raster data (see the VRT driver). This is what is done by the gdal_translate utility for example.

That function will try to validate the creation option list passed to the driver with the **GDALValidateCreationOptions()** method. This check can be disabled by defining the configuration option GDAL_VALIDATE_CREATION_OPTIONS=NO.

After you have finished working with the returned dataset, it is **required** to close it with **GDALClose()**. This does not only close the file handle, but also ensures that all the data and metadata has been written to the dataset (**GDALFlushCache()** is not sufficient for that purpose).

In some situations, the new dataset can be created in another process through the **GDAL API Proxy** mechanism.

Parameters

pszFilename	the name for the new dataset. UTF-8 encoded.
poSrcDS	the dataset being duplicated.
bStrict	TRUE if the copy must be strictly equivalent, or more normally FALSE indicating that the copy may adapt as needed for the output format.
papszOptions	additional format dependent options controlling creation of the output file. The APPEND_SUBDATASET=YES option can be specified to avoid prior destruction of existing dataset.
pfnProgress	a function to be used to report progress of the copy.
pProgressData	application data passed into progress function.

Returns

a pointer to the newly created dataset (may be read-only access).

CPLErr GDALDriver::Delete (const char * pszFilename)

Delete named dataset.

The driver will attempt to delete the named dataset in a driver specific fashion. Full featured drivers will delete all associated files, database objects, or whatever is appropriate. The default behaviour when no driver specific behaviour is provided is to attempt to delete all the files that are returned by GDALGetFileList() on the dataset handle.

It is unwise to have open dataset handles on this dataset when it is deleted.

Equivalent of the C function GDALDeleteDataset().

Parameters

pszFilename name of dataset to delete.

Returns

CE_None on success, or CE_Failure if the operation fails.

static GDALDriver* GDALDriver::FromHandle (GDALDriverH hDriver) inline static

Convert a GDALDriverH to a GDALDriver*.

Since

GDAL 2.3

CPLErr GDALDriver::QuietDelete (const char * pszName) static

Delete dataset if found.

This is a helper method primarily used by Create() and CreateCopy() to predelete any dataset of the name soon to be created. It will attempt to delete the named dataset if one is found, otherwise it does nothing. An error is only returned if the dataset is found but the delete fails.

This is a static method and it doesn't matter what driver instance it is invoked on. It will attempt to discover the correct driver using Identify().

Parameters

pszName the dataset name to try and delete.

Returns

CE_None if the dataset does not exist, or is deleted without issues.

CPLErr GDALDriver::Rename (const char * pszNewName,
const char * pszOldName
)

Rename a dataset.

Rename a dataset. This may including moving the dataset to a new directory or even a new filesystem.

It is unwise to have open dataset handles on this dataset when it is being renamed.

Equivalent of the C function **GDALRenameDataset()**.

Parameters

pszNewName new name for the dataset.

pszOldName old name for the dataset.

Returns

CE_None on success, or CE_Failure if the operation fails.

CPLErr GDALDriver::SetMetadataItem (const char * pszName,
const char * pszValue,
const char * pszDomain = ""
) override virtual

Set single metadata item.

The C function **GDALSetMetadataItem()** does the same thing as this method.

Parameters

pszName the key for the metadata item to fetch.

pszValue the value to assign to the key.

pszDomain the domain to set within, use NULL for the default domain.

Returns

CE_None on success, or an error code on failure.

Reimplemented from **GDALMajorObject**.

static GDALDriverH GDALDriver::ToHandle (GDALDriver * poDriver) inline static

Convert a GDALDriver* to a GDALDriverH.

Since

GDAL 2.3

The documentation for this class was generated from the following files:

- **gdal_priv.h**
- gdaldriver.cpp
