Reference Manual

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 OGRDataSource Class Reference 5

3.1.1 Detailed Description 6

3.1.2 Member Function Documentation 6

3.1.2.1 GetName 6

3.1.2.2 GetLayerCount 6

3.1.2.3 GetLayer 7

3.1.2.4 GetLayerByName 7

3.1.2.5 DeleteLayer 7

3.1.2.6 TestCapability 7

3.1.2.7 CreateLayer 8

3.1.2.8 GetStyleTable 9

3.1.2.9 SetStyleTableDirectly 9

3.1.2.10 SetStyleTable 9

3.1.2.11 ExecuteSQL 9

3.1.2.12 ReleaseResultSet 10

3.1.2.13 SyncToDisk 10

3.1.2.14 Reference 10

3.1.2.15 Dereference 11

3.1.2.16 GetRefCount 11

3.1.2.17 GetSummaryRefCount 11

3.1.2.18 Release 11

3.1.2.19 GetDriver 11

3.1.2.20 SetDriver . . . 12
3.2 OGRLayer Class Reference . . . 13
3.2.1 Detailed Description . . . 14
3.2.2 Member Function Documentation . . . 14
3.2.2.1 GetSpatialFilter . . . 14
3.2.2.2 SetSpatialFilter . . . 14
3.2.2.3 SetSpatialFilterRect . . . 15
3.2.2.4 SetAttributeFilter . . . 15
3.2.2.5 ResetReading . . . 15
3.2.2.6 GetNextFeature . . . 16
3.2.2.7 SetNextByIndex . . . 16
3.2.2.8 GetFeature . . . 16
3.2.2.9 SetFeature . . . 17
3.2.2.10 CreateFeature . . . 17
3.2.2.11 DeleteFeature . . . 17
3.2.2.12 GetLayerDefn . . . 18
3.2.2.13 GetSpatialRef . . . 18
3.2.2.14 GetFeatureCount . . . 18
3.2.2.15 GetExtent . . . 19
3.2.2.16 TestCapability . . . 19
3.2.2.17 GetInfo . . . 20
3.2.2.18 CreateField . . . 20
3.2.2.19 SyncToDisk . . . 21
3.2.2.20 GetStyleTable . . . 21
3.2.2.21 SetStyleTableDirectly . . . 21
3.2.2.22 SetStyleTable . . . 21
3.2.2.23 GetFIDColumn . . . 22
3.2.2.24 GetGeometryColumn . . . 22
3.2.2.25 Reference . . . 22
3.2.2.26 Dereference . . . 22
3.2.2.27 GetRefCount . . . 22
3.3 OGRSFDriver Class Reference . . . 23
3.3.1 Detailed Description . . . 23
3.3.2 Member Function Documentation . . . 23
3.3.2.1 GetName . . . 23
3.3.2.2 Open . . . 23

3.3.2.3 TestCapability . . . 24
3.3.2.4 CreateDataSource . . . 24
3.3.2.5 DeleteDataSource . . . 25
3.4 OGRSFDriverRegistrar Class Reference . . . 26
3.4.1 Detailed Description . . . 26
3.4.2 Member Function Documentation . . . 26
3.4.2.1 GetRegistrar . . . 26
3.4.2.2 Open . . . 26
3.4.2.3 RegisterDriver . . . 27
3.4.2.4 GetDriverCount . . . 27
3.4.2.5 GetDriver . . . 28

4 File Documentation **29**
4.1 ogrsf_frmts.h File Reference . . . 29
4.1.1 Detailed Description . . . 30
4.1.2 Function Documentation . . . 30
4.1.2.1 OGRRegisterAll . . . 30

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

OGRDataSource 5
OGRLayer 13
OGRSFDriver 23
OGRSFDriverRegistrar 26

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

ogr_attrind.h . . . . . **??**
ogrsf_frmts.h 29

Chapter 3

Class Documentation

3.1 OGRDataSource Class Reference

`#include <ogrsf_frmts.h>`

Public Member Functions

- virtual const char * GetName ()=0
- virtual int GetLayerCount ()=0
- virtual OGRLayer * GetLayer (int)=0
- virtual OGRLayer * GetLayerByName (const char *)
- virtual OGRErr DeleteLayer (int)
- virtual int TestCapability (const char *)=0
- virtual OGRLayer * CreateLayer (const char *pszName, OGRSpatialReference *poSpatialRef=NULL, OGRwkbGeometryType eGType=wkbUnknown, char **papszOptions=NULL)
- virtual OGRLayer * **CopyLayer** (OGRLayer *poSrcLayer, const char *pszNewName, char **papszOptions=NULL)
- OGRStyleTable * GetStyleTable ()
- void SetStyleTableDirectly (OGRStyleTable *poStyleTable)
- void SetStyleTable (OGRStyleTable *poStyleTable)
- virtual OGRLayer * ExecuteSQL (const char *pszStatement, OGRGeometry *poSpatialFilter, const char *pszDialect)
- virtual void ReleaseResultSet (OGRLayer *poResultsSet)
- virtual OGRErr SyncToDisk ()
- int Reference ()
- int Dereference ()
- int GetRefCount () const
- int GetSummaryRefCount () const
- OGRErr Release ()
- OGRSFDriver * GetDriver () const
- void SetDriver (OGRSFDriver *poDriver)

Static Public Member Functions

- static void **DestroyDataSource** (OGRDataSource *)

Protected Member Functions

- OGRErr **ProcessSQLCreateIndex** (const char *)
- OGRErr **ProcessSQLDropIndex** (const char *)

Protected Attributes

- OGRStyleTable * **m_poStyleTable**
- int **m_nRefCount**
- OGRSFDriver * **m_poDriver**

Friends

- class OGRSFDriverRegistrar

3.1.1 Detailed Description

This class represents a data source. A data source potentially consists of many layers (OGRLayer). A data source normally consists of one, or a related set of files, though the name doesn't have to be a real item in the file system.

When an OGRDataSource is destroyed, all it's associated OGRLayers objects are also destroyed.

3.1.2 Member Function Documentation

3.1.2.1 const char * OGRDataSource::GetName () [pure virtual]

Returns the name of the data source. This string should be sufficient to open the data source if passed to the same OGRSFDriver that this data source was opened with, but it need not be exactly the same string that was used to open the data source. Normally this is a filename.

This method is the same as the C function OGR_DS_GetName().

Returns:

pointer to an internal name string which should not be modified or freed by the caller.

3.1.2.2 int OGRDataSource::GetLayerCount () [pure virtual]

Get the number of layers in this data source.

This method is the same as the C function OGR_DS_GetLayerCount().

Returns:

layer count.

3.1.2.3 OGRLayer * OGRDataSource::GetLayer (int *iLayer*) [pure virtual]

Fetch a layer by index. The returned layer remains owned by the OGRDataSource and should not be deleted by the application.

This method is the same as the C function OGR_DS_GetLayer().

Parameters:

iLayer a layer number between 0 and GetLayerCount()-1.

Returns:

the layer, or NULL if iLayer is out of range or an error occurs.

3.1.2.4 OGRLayer * OGRDataSource::GetLayerByName (const char * *pszLayerName*) [virtual]

Fetch a layer by name. The returned layer remains owned by the OGRDataSource and should not be deleted by the application.

This method is the same as the C function OGR_DS_GetLayerByName().

Parameters:

pszLayerName the layer name of the layer to fetch.

Returns:

the layer, or NULL if Layer is not found or an error occurs.

3.1.2.5 OGRErr OGRDataSource::DeleteLayer (int *iLayer*) [virtual]

Delete the indicated layer from the datasource. If this method is supported the ODsCDeleteLayer capability will test TRUE on the OGRDataSource.

This method is the same as the C function OGR_DS_DeleteLayer().

Parameters:

iLayer the index of the layer to delete.

Returns:

OGRERR_NONE on success, or OGRERR_UNSUPPORTED_OPERATION if deleting layers is not supported for this datasource.

3.1.2.6 int OGRDataSource::TestCapability (const char * *pszCapability*) [pure virtual]

Test if capability is available.

One of the following data source capability names can be passed into this method, and a TRUE or FALSE value will be returned indicating whether or not the capability is available for this object.

- **ODsCCreateLayer**: True if this datasource can create new layers.

The #define macro forms of the capability names should be used in preference to the strings themselves to avoid mispelling.

This method is the same as the C function OGR_DS_TestCapability().

Parameters:

pszCapability the capability to test.

Returns:

TRUE if capability available otherwise FALSE.

3.1.2.7 OGRLayer * OGRDataSource::CreateLayer (const char * *pszName*, OGRSpatialReference * *poSpatialRef* = NULL, OGRwkbGeometryType *eGType* = wkbUnknown, char ** *papszOptions* = NULL) [virtual]

This method attempts to create a new layer on the data source with the indicated name, coordinate system, geometry type. The papszOptions argument can be used to control driver specific creation options. These options are normally documented in the format specific documentation.

Parameters:

pszName the name for the new layer. This should ideally not match any existing layer on the data-source.

poSpatialRef the coordinate system to use for the new layer, or NULL if no coordinate system is available.

eGType the geometry type for the layer. Use wkbUnknown if there are no constraints on the types geometry to be written.

papszOptions a StringList of name=value options. Options are driver specific.

Returns:

NULL is returned on failure, or a new OGRLayer handle on success.

Example:

```
#include "ogrsf_frmts.h"
#include "cpl_string.h"

...

        OGRLayer *poLayer;
        char     *papszOptions;

        if( !poDS->TestCapability( ODsCCreateLayer ) )
        {
            ...
        }

        papszOptions = CSLSetNameValue( papszOptions, "DIM", "2" );
        poLayer = poDS->CreateLayer( "NewLayer", NULL, wkbUnknown,
                                     papszOptions );
        CSLDestroy( papszOptions );
```

```
if( poLayer == NULL )
{
    ...
}
```

3.1.2.8 void OGRDataSource::GetStyleTable () [inline]

Returns data source style table.

This method is the same as the C function OGR_DS_GetStyleTable().

Returns:

pointer to a style table which should not be modified or freed by the caller.

3.1.2.9 void OGRDataSource::SetStyleTableDirectly (OGRStyleTable * *poStyleTable*) [inline]

Set data source style table.

This method operate exactly as OGRDataSource::SetStyleTable() except that it assumes ownership of the passed table.

This method is the same as the C function OGR_DS_SetStyleTableDirectly().

Parameters:

poStyleTable pointer to style table to set

3.1.2.10 void OGRDataSource::SetStyleTable (OGRStyleTable * *poStyleTable*) [inline]

Set data source style table.

This method operate exactly as OGRDataSource::SetStyleTableDirectly() except that it does not assume ownership of the passed table.

This method is the same as the C function OGR_DS_SetStyleTable().

Parameters:

poStyleTable pointer to style table to set

3.1.2.11 OGRLayer * OGRDataSource::ExecuteSQL (const char * *pszStatement*, OGRGeometry * *poSpatialFilter*, const char * *pszDialect*) [virtual]

Execute an SQL statement against the data store.

The result of an SQL query is either NULL for statements that are in error, or that have no results set, or an OGRLayer pointer representing a results set from the query. Note that this OGRLayer is in addition to the layers in the data store and must be destroyed with OGRDataSource::ReleaseResultsSet() before the data source is closed (destroyed).

This method is the same as the C function OGR_DS_ExecuteSQL().

For more information on the SQL dialect supported internally by OGR review the OGR SQL document. Some drivers (ie. Oracle and PostGIS) pass the SQL directly through to the underlying RDBMS.

Parameters:

pszStatement the SQL statement to execute.

poSpatialFilter geometry which represents a spatial filter.

pszDialect allows control of the statement dialect. By default it is assumed to be "generic" SQL, whatever that is.

Returns:

an OGRLayer containing the results of the query. Deallocate with ReleaseResultsSet().

3.1.2.12 void OGRDataSource::ReleaseResultSet (OGRLayer * *poResultsSet*) [virtual]

Release results of ExecuteSQL().

This method should only be used to deallocate OGRLayers resulting from an ExecuteSQL() call on the same OGRDataSource. Failure to deallocate a results set before destroying the OGRDataSource may cause errors.

This method is the same as the C function OGR_L_ReleaseResultsSet().

Parameters:

poResultsSet the result of a previous ExecuteSQL() call.

3.1.2.13 OGRErr OGRDataSource::SyncToDisk () [virtual]

Flush pending changes to disk.

This call is intended to force the datasource to flush any pending writes to disk, and leave the disk file in a consistent state. It would not normally have any effect on read-only datasources.

Some data sources do not implement this method, and will still return OGRERR_NONE. An error is only returned if an error occurs while attempting to flush to disk.

The default implementation of this method just calls the SyncToDisk() method on each of the layers. Conceptionally, calling SyncToDisk() on a datasource should include any work that might be accomplished by calling SyncToDisk() on layers in that data source.

This method is the same as the C function OGR_DS_SyncToDisk().

Returns:

OGRERR_NONE if no error occurs (even if nothing is done) or an error code.

3.1.2.14 int OGRDataSource::Reference ()

Increment datasource reference count.

This method is the same as the C function OGR_DS_Reference().

Returns:

the reference count after incrementing.

3.1.2.15 int OGRDataSource::Dereference ()

Decrement datasource reference count.

This method is the same as the C function OGR_DS_Dereference().

Returns:

the reference count after decrementing.

3.1.2.16 int OGRDataSource::GetRefCount () const

Fetch reference count.

This method is the same as the C function OGR_DS_GetRefCount().

Returns:

the current reference count for the datasource object itself.

3.1.2.17 int OGRDataSource::GetSummaryRefCount () const

Fetch reference count of datasource and all owned layers.

This method is the same as the C function OGR_DS_GetSummaryRefCount().

Returns:

the current summary reference count for the datasource and its layers.

3.1.2.18 OGRErr OGRDataSource::Release ()

Drop a reference to this datasource, and if the reference count drops to zero close (destroy) the datasource. Internally this actually calls the OGRSFDriverRegistrary::ReleaseDataSource() method. This method is essentially a convenient alias.

This method is the same as the C function OGRReleaseDataSource().

Returns:

OGRERR_NONE on success or an error code.

3.1.2.19 OGRSFDriver * OGRDataSource::GetDriver () const

Returns the driver that the dataset was opened with.

This method is the same as the C function OGR_DS_GetDriver().

Returns:

NULL if driver info is not available, or pointer to a driver owned by the OGRSFDriverManager.

3.1.2.20 void OGRDataSource::SetDriver (OGRSFDriver * *poDriver*)

Sets the driver that the dataset was created or opened with.

Note:

This method is not exposed as the OGR C API function.

Parameters:

poDriver pointer to driver instance associated with the data source.

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

- ogrsf_frmts.h
- ogrsf_frmts.dox

3.2 OGRLayer Class Reference

`#include <ogrsf_frmts.h>`

Public Member Functions

- virtual OGRGeometry * GetSpatialFilter ()
- virtual void SetSpatialFilter (OGRGeometry *)
- virtual void SetSpatialFilterRect (double dfMinX, double dfMinY, double dfMaxX, double dfMaxY)
- virtual OGRErr SetAttributeFilter (const char *)
- virtual void ResetReading ()=0
- virtual OGRFeature * GetNextFeature ()=0
- virtual OGRErr SetNextByIndex (long nIndex)
- virtual OGRFeature * GetFeature (long nFID)
- virtual OGRErr SetFeature (OGRFeature *poFeature)
- virtual OGRErr CreateFeature (OGRFeature *poFeature)
- virtual OGRErr DeleteFeature (long nFID)
- virtual OGRFeatureDefn * GetLayerDefn ()=0
- virtual OGRSpatialReference * GetSpatialRef ()
- virtual int GetFeatureCount (int bForce=TRUE)
- virtual OGRErr GetExtent (OGREnvelope *psExtent, int bForce=TRUE)
- virtual int TestCapability (const char *)=0
- virtual const char * GetInfo (const char *)
- virtual OGRErr CreateField (OGRFieldDefn *poField, int bApproxOK=TRUE)
- virtual OGRErr SyncToDisk ()
- OGRStyleTable * GetStyleTable ()
- void SetStyleTableDirectly (OGRStyleTable *poStyleTable)
- void SetStyleTable (OGRStyleTable *poStyleTable)
- virtual OGRErr **StartTransaction** ()
- virtual OGRErr **CommitTransaction** ()
- virtual OGRErr **RollbackTransaction** ()
- virtual const char * GetFIDColumn ()
- virtual const char * GetGeometryColumn ()
- int Reference ()
- int Dereference ()
- int GetRefCount () const
- GIntBig **GetFeaturesRead** ()
- OGRErr **InitializeIndexSupport** (const char *)
- OGRLayerAttrIndex * **GetIndex** ()

Protected Member Functions

- int **FilterGeometry** (OGRGeometry *)
- int **InstallFilter** (OGRGeometry *)

Protected Attributes

- int **m_bFilterIsEnvelope**
- OGRGeometry * **m_poFilterGeom**
- OGREnvelope **m_sFilterEnvelope**
- OGRStyleTable * **m_poStyleTable**
- OGRFeatureQuery * **m_poAttrQuery**
- OGRLayerAttrIndex * **m_poAttrIndex**
- int **m_nRefCount**
- GIntBig **m_nFeaturesRead**

3.2.1 Detailed Description

This class represents a layer of simple features, with access methods.

3.2.2 Member Function Documentation

3.2.2.1 OGRGeometry * OGRLayer::GetSpatialFilter () [virtual]

This method returns the current spatial filter for this layer.

The returned pointer is to an internally owned object, and should not be altered or deleted by the caller.

This method is the same as the C function OGR_L_GetSpatialFilter().

Returns:

spatial filter geometry.

3.2.2.2 void OGRLayer::SetSpatialFilter (OGRGeometry * *poFilter*) [virtual]

Set a new spatial filter.

This method set the geometry to be used as a spatial filter when fetching features via the GetNextFeature() method. Only features that geometrically intersect the filter geometry will be returned.

Currently this test is may be inaccurately implemented, but it is guaranteed that all features who's envelope (as returned by OGRGeometry::getEnvelope()) overlaps the envelope of the spatial filter will be returned. This can result in more shapes being returned that should strictly be the case.

This method makes an internal copy of the passed geometry. The passed geometry remains the responsibility of the caller, and may be safely destroyed.

For the time being the passed filter geometry should be in the same SRS as the layer (as returned by OGRLayer::GetSpatialRef()). In the future this may be generalized.

This method is the same as the C function OGR_L_SetSpatialFilter().

Parameters:

poFilter the geometry to use as a filtering region. NULL may be passed indicating that the current spatial filter should be cleared, but no new one instituted.

3.2.2.3 void OGRLayer::SetSpatialFilterRect (double *dfMinX*, double *dfMinY*, double *dfMaxX*, double *dfMaxY*) [virtual]

Set a new rectangular spatial filter.

This method set rectangle to be used as a spatial filter when fetching features via the GetNextFeature() method. Only features that geometrically intersect the given rectangle will be returned.

The x/y values should be in the same coordinate system as the layer as a whole (as returned by OGRLayer::GetSpatialRef()). Internally this method is normally implemented as creating a 5 vertex closed rectangular polygon and passing it to OGRLayer::SetSpatialFilter(). It exists as a convenience.

The only way to clear a spatial filter set with this method is to call OGRLayer::SetSpatialFilter(NULL).

This method is the same as the C function OGR_L_SetSpatialFilterRect().

Parameters:

dfMinX the minimum X coordinate for the rectangular region.

dfMinY the minimum Y coordinate for the rectangular region.

dfMaxX the maximum X coordinate for the rectangular region.

dfMaxY the maximum Y coordinate for the rectangular region.

3.2.2.4 void OGRLayer::SetAttributeFilter (const char * *pszQuery*) [virtual]

Set a new attribute query.

This method sets the attribute query string to be used when fetching features via the GetNextFeature() method. Only features for which the query evaluates as true will be returned.

The query string should be in the format of an SQL WHERE clause. For instance "population > 1000000 and population < 5000000" where population is an attribute in the layer. The query format is a restricted form of SQL WHERE clause as defined "eq_format=restricted_where" about half way through this document:

http://ogdi.sourceforge.net/prop/6.2.CapabilitiesMetadata.html

Note that installing a query string will generally result in resetting the current reading position (ala ResetReading()).

This method is the same as the C function OGR_L_SetAttributeFilter().

Parameters:

pszQuery query in restricted SQL WHERE format, or NULL to clear the current query.

Returns:

OGRERR_NONE if successfully installed, or an error code if the query expression is in error, or some other failure occurs.

3.2.2.5 void OGRLayer::ResetReading () [pure virtual]

Reset feature reading to start on the first feature. This affects GetNextFeature().

This method is the same as the C function OGR_L_ResetReading().

3.2.2.6 OGRFeature * OGRLayer::GetNextFeature () [pure virtual]

Fetch the next available feature from this layer. The returned feature becomes the responsiblity of the caller to delete.

Only features matching the current spatial filter (set with SetSpatialFilter()) will be returned.

This method implements sequential access to the features of a layer. The ResetReading() method can be used to start at the beginning again.

This method is the same as the C function OGR_L_GetNextFeature().

Returns:

a feature, or NULL if no more features are available.

3.2.2.7 OGRErr OGRLayer::SetNextByIndex (long *nIndex*) [virtual]

Move read cursor to the nIndex'th feature in the current resultset.

This method allows positioning of a layer such that the GetNextFeature() call will read the requested feature, where nIndex is an absolute index into the current result set. So, setting it to 3 would mean the next feature read with GetNextFeature() would have been the 4th feature to have been read if sequential reading took place from the beginning of the layer, including accounting for spatial and attribute filters.

Only in rare circumstances is SetNextByIndex() efficiently implemented. In all other cases the default implementation which calls ResetReading() and then calls GetNextFeature() nIndex times is used. To determine if fast seeking is available on the current layer use the TestCapability() method with a value of OLCFastSetNextByIndex.

This method is the same as the C function OGR_L_SetNextByIndex().

Parameters:

nIndex the index indicating how many steps into the result set to seek.

Returns:

OGRERR_NONE on success or an error code.

3.2.2.8 OGRFeature * OGRLayer::GetFeature (long *nFID*) [virtual]

Fetch a feature by it's identifier.

This function will attempt to read the identified feature. The nFID value cannot be OGRNullFID. Success or failure of this operation is unaffected by the spatial or attribute filters.

If this method returns a non-NULL feature, it is guaranteed that it's feature id (OGRFeature::GetFID()) will be the same as nFID.

Use OGRLayer::TestCapability(OLCRandomRead) to establish if this layer supports efficient random access reading via GetFeature(); however, the call should always work if the feature exists as a fallback implementation just scans all the features in the layer looking for the desired feature.

Sequential reads are generally considered interrupted by a GetFeature() call.

This method is the same as the C function OGR_L_GetFeature().

Parameters:

nFID the feature id of the feature to read.

Returns:

a feature now owned by the caller, or NULL on failure.

3.2.2.9 OGRErr OGRLayer::SetFeature (OGRFeature * *poFeature*) [virtual]

Rewrite an existing feature.

This method will write a feature to the layer, based on the feature id within the OGRFeature.

Use OGRLayer::TestCapability(OLCRandomWrite) to establish if this layer supports random access writing via SetFeature().

This method is the same as the C function OGR_L_SetFeature().

Parameters:

poFeature the feature to write.

Returns:

OGRERR_NONE if the operation works, otherwise an appropriate error code.

3.2.2.10 OGRErr OGRLayer::CreateFeature (OGRFeature * *poFeature*) [virtual]

Create and write a new feature within a layer.

The passed feature is written to the layer as a new feature, rather than overwriting an existing one. If the feature has a feature id other than OGRNullFID, then the native implementation may use that as the feature id of the new feature, but not necessarily. Upon successful return the passed feature will have been updated with the new feature id.

This method is the same as the C function OGR_L_CreateFeature().

Parameters:

poFeature the feature to write to disk.

Returns:

OGRERR_NONE on success.

3.2.2.11 OGRErr OGRLayer::DeleteFeature (long *nFID*) [virtual]

Delete feature from layer.

The feature with the indicated feature id is deleted from the layer if supported by the driver. Most drivers do not support feature deletion, and will return OGRERR_UNSUPPORTED_OPERATION. The TestCapability() layer method may be called with OLCDeleteFeature to check if the driver supports feature deletion.

This method is the same as the C function OGR_L_DeleteFeature().

Parameters:

nFID the feature id to be deleted from the layer

Returns:

OGRERR_NONE on success.

3.2.2.12 OGRFeatureDefn * OGRLayer::GetLayerDefn () [pure virtual]

Fetch the schema information for this layer.

The returned OGRFeatureDefn is owned by the OGRLayer, and should not be modified or freed by the application. It encapsulates the attribute schema of the features of the layer.

This method is the same as the C function OGR_L_GetLayerDefn().

Returns:

feature definition.

3.2.2.13 OGRSpatialReference * OGRLayer::GetSpatialRef () [inline, virtual]

Fetch the spatial reference system for this layer.

The returned object is owned by the OGRLayer and should not be modified or freed by the application.

This method is the same as the C function OGR_L_GetSpatialRef().

Returns:

spatial reference, or NULL if there isn't one.

3.2.2.14 int OGRLayer::GetFeatureCount (int *bForce* = TRUE) [virtual]

Fetch the feature count in this layer.

Returns the number of features in the layer. For dynamic databases the count may not be exact. If bForce is FALSE, and it would be expensive to establish the feature count a value of -1 may be returned indicating that the count isn't know. If bForce is TRUE some implementations will actually scan the entire layer once to count objects.

The returned count takes the spatial filter into account.

This method is the same as the C function OGR_L_GetFeatureCount().

Parameters:

bForce Flag indicating whether the count should be computed even if it is expensive.

Returns:

feature count, -1 if count not known.

3.2.2.15 OGRErr OGRLayer::GetExtent (OGREnvelope * *psExtent*, int *bForce* = TRUE) [virtual]

Fetch the extent of this layer.

Returns the extent (MBR) of the data in the layer. If bForce is FALSE, and it would be expensive to establish the extent then OGRERR_FAILURE will be returned indicating that the extent isn't know. If bForce is TRUE then some implementations will actually scan the entire layer once to compute the MBR of all the features in the layer.

Depending on the drivers, the returned extent may or may not take the spatial filter into account. So it is safer to call GetExtent() without setting a spatial filter.

Layers without any geometry may return OGRERR_FAILURE just indicating that no meaningful extents could be collected.

This method is the same as the C function OGR_L_GetExtent().

Parameters:

psExtent the structure in which the extent value will be returned.

bForce Flag indicating whether the extent should be computed even if it is expensive.

Returns:

OGRERR_NONE on success, OGRERR_FAILURE if extent not known.

3.2.2.16 int OGRLayer::TestCapability (const char * *pszCap*) [pure virtual]

Test if this layer supported the named capability.

The capability codes that can be tested are represented as strings, but #defined constants exists to ensure correct spelling. Specific layer types may implement class specific capabilities, but this can't generally be discovered by the caller.

- **OLCRandomRead** / "RandomRead": TRUE if the GetFeature() method is implemented in an optimized way for this layer, as opposed to the default implementation using ResetReading() and GetNextFeature() to find the requested feature id.
- **OLCSequentialWrite** / "SequentialWrite": TRUE if the CreateFeature() method works for this layer. Note this means that this particular layer is writable. The same OGRLayer class may returned FALSE for other layer instances that are effectively read-only.
- **OLCRandomWrite** / "RandomWrite": TRUE if the SetFeature() method is operational on this layer. Note this means that this particular layer is writable. The same OGRLayer class may returned FALSE for other layer instances that are effectively read-only.
- **OLCFastSpatialFilter** / "FastSpatialFilter": TRUE if this layer implements spatial filtering efficiently. Layers that effectively read all features, and test them with the OGRFeature intersection methods should return FALSE. This can be used as a clue by the application whether it should build and maintain it's own spatial index for features in this layer.
- **OLCFastFeatureCount** / "FastFeatureCount": TRUE if this layer can return a feature count (via OGRLayer::GetFeatureCount()) efficiently ... ie. without counting the features. In some cases this will return TRUE until a spatial filter is installed after which it will return FALSE.

- **OLCFastGetExtent** / "FastGetExtent": TRUE if this layer can return its data extent (via OGRLayer::GetExtent()) efficiently ... ie. without scanning all the features. In some cases this will return TRUE until a spatial filter is installed after which it will return FALSE.
- **OLCFastSetNextByIndex** / "FastSetNextByIndex": TRUE if this layer can perform the SetNextByIndex() call efficiently, otherwise FALSE.

This method is the same as the C function OGR_L_TestCapability().

Parameters:

pszCap the name of the capability to test.

Returns:

TRUE if the layer has the requested capability, or FALSE otherwise. OGRLayers will return FALSE for any unrecognised capabilities.

3.2.2.17 const char * OGRLayer::GetInfo (const char * *pszTag*) [virtual]

Fetch metadata from layer.

This method can be used to fetch various kinds of metadata or layer specific information encoded as a string. It is anticipated that various tag values will be defined with well known semantics, while other tags will be used for driver/application specific purposes.

This method is deprecated and will be replaced with a more general metadata model in the future. At this time no drivers return information via the GetInfo() call.

Parameters:

pszTag the tag for which information is being requested.

Returns:

the value of the requested tag, or NULL if that tag does not have a value, or is unknown.

3.2.2.18 OGRErr OGRLayer::CreateField (OGRFieldDefn * *poField*, int *bApproxOK* = TRUE) [virtual]

Create a new field on a layer. You must use this to create new fields on a real layer. Internally the OGRFeatureDefn for the layer will be updated to reflect the new field. Applications should never modify the OGRFeatureDefn used by a layer directly.

This function is the same as the C function OGR_L_CreateField().

Parameters:

poField field definition to write to disk.

bApproxOK If TRUE, the field may be created in a slightly different form depending on the limitations of the format driver.

Returns:

OGRERR_NONE on success.

3.2.2.19 OGRErr OGRLayer::SyncToDisk () [virtual]

Flush pending changes to disk.

This call is intended to force the layer to flush any pending writes to disk, and leave the disk file in a consistent state. It would not normally have any effect on read-only datasources.

Some layers do not implement this method, and will still return OGRERR_NONE. The default implementation just returns OGRERR_NONE. An error is only returned if an error occurs while attempting to flush to disk.

This method is the same as the C function OGR_L_SyncToDisk().

Returns:

OGRERR_NONE if no error occurs (even if nothing is done) or an error code.

3.2.2.20 void OGRLayer::GetStyleTable () [inline]

Returns layer style table.

This method is the same as the C function OGR_L_GetStyleTable().

Returns:

pointer to a style table which should not be modified or freed by the caller.

3.2.2.21 void OGRLayer::SetStyleTableDirectly (OGRStyleTable * *poStyleTable*) [inline]

Set layer style table.

This method operate exactly as OGRLayer::SetStyleTable() except that it assumes ownership of the passed table.

This method is the same as the C function OGR_L_SetStyleTableDirectly().

Parameters:

poStyleTable pointer to style table to set

3.2.2.22 void OGRLayer::SetStyleTable (OGRStyleTable * *poStyleTable*) [inline]

Set layer style table.

This method operate exactly as OGRLayer::SetStyleTableDirectly() except that it does not assume ownership of the passed table.

This method is the same as the C function OGR_L_SetStyleTable().

Parameters:

poStyleTable pointer to style table to set

3.2.2.23 const char * OGRLayer::GetFIDColumn () [virtual]

This method returns the name of the underlying database column being used as the FID column, or "" if not supported.

This method is the same as the C function OGR_L_GetFIDColumn().

Returns:

fid column name.

3.2.2.24 const char * OGRLayer::GetGeometryColumn () [virtual]

This method returns the name of the underlying database column being used as the geometry column, or "" if not supported.

This method is the same as the C function OGR_L_GetFIDColumn().

Returns:

fid column name.

3.2.2.25 int OGRLayer::Reference ()

Increment layer reference count.

This method is the same as the C function OGR_L_Reference().

Returns:

the reference count after incrementing.

3.2.2.26 int OGRLayer::Dereference ()

Decrement layer reference count.

This method is the same as the C function OGR_L_Dereference().

Returns:

the reference count after decrementing.

3.2.2.27 int OGRLayer::GetRefCount () const

Fetch reference count.

This method is the same as the C function OGR_L_GetRefCount().

Returns:

the current reference count for the layer object itself.

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

- ogrsf_frmts.h
- ogrsf_frmts.dox

3.3 OGRSFDriver Class Reference

`#include <ogrsf_frmts.h>`

Public Member Functions

- virtual const char * GetName ()=0
- virtual OGRDataSource * Open (const char *pszName, int bUpdate=FALSE)=0
- virtual int TestCapability (const char *)=0
- virtual OGRDataSource * CreateDataSource (const char *pszName, char **=NULL)
- virtual OGRErr DeleteDataSource (const char *pszName)
- virtual OGRDataSource * **CopyDataSource** (OGRDataSource *poSrcDS, const char *pszNewName, char **papszOptions=NULL)

3.3.1 Detailed Description

Represents an operational format driver.

One OGRSFDriver derived class will normally exist for each file format registered for use, regardless of whether a file has or will be opened. The list of available drivers is normally managed by the OGRSFDriverRegistrar.

3.3.2 Member Function Documentation

3.3.2.1 const char * OGRSFDriver::GetName () `[pure virtual]`

Fetch name of driver (file format). This name should be relatively short (10-40 characters), and should reflect the underlying file format. For instance "ESRI Shapefile".

This method is the same as the C function OGR_Dr_GetName().

Returns:

driver name. This is an internal string and should not be modified or freed.

3.3.2.2 OGRDataSource * OGRSFDriver::Open (const char * *pszName*, int *bUpdate* = `FALSE`) `[pure virtual]`

Attempt to open file with this driver.

This method is what OGRSFDriverRegistrar uses to implement its Open() method. See it for more details.

Note, drivers do not normally set their own m_poDriver value, so a direct call to this method (instead of indirectly via OGRSFDriverRegistrar) will usually result in a datasource that does not know what driver it relates to if GetDriver() is called on the datasource. The application may directly call SetDriver() after opening with this method to avoid this problem.

This method is the same as the C function OGR_Dr_Open().

Parameters:

pszName the name of the file, or data source to try and open.

bUpdate TRUE if update access is required, otherwise FALSE (the default).

Returns:

NULL on error or if the pass name is not supported by this driver, otherwise a pointer to an OGRDataSource. This OGRDataSource should be closed by deleting the object when it is no longer needed.

3.3.2.3 int OGRSFDriver::TestCapability (const char * *pszCapability*) [pure virtual]

Test if capability is available.

One of the following data source capability names can be passed into this method, and a TRUE or FALSE value will be returned indicating whether or not the capability is available for this object.

- **ODrCCreateDataSource**: True if this driver can support creating data sources.
- **ODrCDeleteDataSource**: True if this driver supports deleting data sources.

The #define macro forms of the capability names should be used in preference to the strings themselves to avoid mispelling.

This method is the same as the C function OGR_Dr_TestCapability().

Parameters:

pszCapability the capability to test.

Returns:

TRUE if capability available otherwise FALSE.

3.3.2.4 OGRDataSource * OGRSFDriver::CreateDataSource (const char * *pszName*, char ** *papszOptions* = NULL) [virtual]

This method attempts to create a new data source based on the passed driver. The papszOptions argument can be used to control driver specific creation options. These options are normally documented in the format specific documentation.

This method is the same as the C function OGR_Dr_CreateDataSource().

Note:

This method does **NOT** attach driver instance to the returned data source, so caller should expect that OGRDataSource::GetDriver() will return NULL pointer. In order to attach driver to the returned data soruce, it is required to use C function OGR_Dr_CreateDataSource. This behavior is related to fix of issue reported in Ticket #1233.

Parameters:

pszName the name for the new data source.

papszOptions a StringList of name=value options. Options are driver specific, and driver information can be found at the following url: http://www.gdal.org/ogr/ogr_formats.html

Returns:

NULL is returned on failure, or a new OGRDataSource on success.

3.3.2.5 OGRErr OGRSFDriver::DeleteDataSource (const char * *pszDataSource*) [virtual]

Destroy a datasource.

Destroy the named datasource. Normally it would be safest if the datasource was not open at the time.

Whether this is a supported operation on this driver case be tested using TestCapability() on ODrCDeleteDataSource.

This method is the same as the C function OGR_Dr_DeleteDataSource().

Parameters:

pszDataSource the name of the datasource to delete.

Returns:

OGRERR_NONE on success, and OGRERR_UNSUPPORTED_OPERATION if this is not supported by this driver.

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

- ogrsf_frmts.h
- ogrsf_frmts.dox

3.4 OGRSFDriverRegistrar Class Reference

`#include <ogrsf_frmts.h>`

Public Member Functions

- OGRDataSource * **OpenShared** (const char *pszName, int bUpdate=FALSE, OGRSFDriver **ppoDriver=NULL)
- OGRErr **ReleaseDataSource** (OGRDataSource *)
- void RegisterDriver (OGRSFDriver *poDriver)
- int GetDriverCount (void)
- OGRSFDriver * GetDriver (int iDriver)
- OGRSFDriver * **GetDriverByName** (const char *)
- int **GetOpenDSCount** ()
- OGRDataSource * **GetOpenDS** (int)
- void **AutoLoadDrivers** ()

Static Public Member Functions

- static OGRSFDriverRegistrar * GetRegistrar ()
- static OGRDataSource * Open (const char *pszName, int bUpdate=FALSE, OGRSFDriver **ppoDriver=NULL)

3.4.1 Detailed Description

Singleton manager for OGRSFDriver instances that will be used to try and open datasources. Normally the registrar is populated with standard drivers using the OGRRegisterAll() function and does not need to be directly accessed. The driver registrar and all registered drivers may be cleaned up on shutdown using OGRCleanupAll().

3.4.2 Member Function Documentation

3.4.2.1 OGRSFDriverRegistrar * OGRSFDriverRegistrar::GetRegistrar () [static]

Return the driver manager, creating one if none exist.

Returns:

the driver manager.

3.4.2.2 OGRDataSource * OGRSFDriverRegistrar::Open (const char * *pszName*, int *bUpdate* = FALSE, OGRSFDriver ** *ppoDriver* = NULL) [static]

Open a file / data source with one of the registered drivers.

This method loops through all the drivers registered with the driver manager trying each until one succeeds with the given data source. This method is static. Applications don't normally need to use any other OGRSFDriverRegistrar methods directly, not do they normally need to have a pointer to an OGRSFDriverRegistrar instance.

If this method fails, CPLGetLastErrorMsg() can be used to check if there is an error message explaining why.

This method is the same as the C function OGROpen().

Parameters:

pszName the name of the file, or data source to open.

bUpdate FALSE for read-only access (the default) or TRUE for read-write access.

ppoDriver if non-NULL, this argument will be updated with a pointer to the driver which was used to open the data source.

Returns:

NULL on error or if the pass name is not supported by this driver, otherwise a pointer to an OGRDataSource. This OGRDataSource should be closed by deleting the object when it is no longer needed.

Example:

```
OGRDataSource *poDS;

poDS = OGRSFDriverRegistrar::Open( "polygon.shp" );
if( poDS == NULL )
{
    return;
}

... use the data source ...

delete poDS;
```

3.4.2.3 void OGRSFDriverRegistrar::RegisterDriver (OGRSFDriver * *poDriver*)

Add a driver to the list of registered drivers.

If the passed driver is already registered (based on pointer comparison) then the driver isn't registered. New drivers are added at the end of the list of registered drivers.

This method is the same as the C function OGRRegisterDriver().

Parameters:

poDriver the driver to add.

3.4.2.4 int OGRSFDriverRegistrar::GetDriverCount (void)

Fetch the number of registered drivers.

This method is the same as the C function OGRGetDriverCount().

Returns:

the drivers count.

3.4.2.5 OGRSFDriver * OGRSFDriverRegistrar::GetDriver (int *iDriver*)

Fetch the indicated driver.

This method is the same as the C function OGRGetDriver().

Parameters:

iDriver the driver index, from 0 to GetDriverCount()-1.

Returns:

the driver, or NULL if iDriver is out of range.

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

- ogrsf_frmts.h
- ogrsf_frmts.dox

Chapter 4

File Documentation

4.1 ogrsf_frmts.h File Reference

```
#include "ogr_feature.h"
#include "ogr_featurestyle.h"
```

Classes

- class OGRLayer
- class OGRDataSource
- class OGRSFDriver
- class OGRSFDriverRegistrar

Functions

- CPL_C_START void CPL_DLL OGRRegisterAll ()
- void CPL_DLL **RegisterOGRShape** ()
- void CPL_DLL **RegisterOGRNTF** ()
- void CPL_DLL **RegisterOGRFME** ()
- void CPL_DLL **RegisterOGRSDTS** ()
- void CPL_DLL **RegisterOGRTiger** ()
- void CPL_DLL **RegisterOGRS57** ()
- void CPL_DLL **RegisterOGRTAB** ()
- void CPL_DLL **RegisterOGRMIF** ()
- void CPL_DLL **RegisterOGROGDI** ()
- void CPL_DLL **RegisterOGRODBC** ()
- void CPL_DLL **RegisterOGRPG** ()
- void CPL_DLL **RegisterOGRMySQL** ()
- void CPL_DLL **RegisterOGROCI** ()
- void CPL_DLL **RegisterOGRDGN** ()
- void CPL_DLL **RegisterOGRGML** ()
- void CPL_DLL **RegisterOGRKML** ()
- void CPL_DLL **RegisterOGRGeoJSON** ()
- void CPL_DLL **RegisterOGRAVCBin** ()

- void CPL_DLL **RegisterOGRAVCE00** ()
- void CPL_DLL **RegisterOGRREC** ()
- void CPL_DLL **RegisterOGRMEM** ()
- void CPL_DLL **RegisterOGRVRT** ()
- void CPL_DLL **RegisterOGRDODS** ()
- void CPL_DLL **RegisterOGRSQLite** ()
- void CPL_DLL **RegisterOGRCSV** ()
- void CPL_DLL **RegisterOGRILI1** ()
- void CPL_DLL **RegisterOGRILI2** ()
- void CPL_DLL **RegisterOGRGRASS** ()
- void CPL_DLL **RegisterOGRPGeo** ()
- void CPL_DLL **RegisterOGRDXFDWG** ()
- void CPL_DLL **RegisterOGRSDE** ()
- void CPL_DLL **RegisterOGRIDB** ()
- void CPL_DLL **RegisterOGRGMT** ()
- void CPL_DLL **RegisterOGRBNA** ()
- void CPL_DLL **RegisterOGRGPX** ()
- void CPL_DLL **RegisterOGRGeoconcept** ()
- void CPL_DLL **RegisterOGRIngres** ()
- void CPL_DLL **RegisterOGRXPlane** ()
- void CPL_DLL **RegisterOGRNAS** ()

4.1.1 Detailed Description

Classes related to registration of format support, and opening datasets.

4.1.2 Function Documentation

4.1.2.1 int OGRRegisterAll ()

Register all drivers.

Index

CreateDataSource
    OGRSFDriver, 24
CreateFeature
    OGRLayer, 17
CreateField
    OGRLayer, 20
CreateLayer
    OGRDataSource, 8

DeleteDataSource
    OGRSFDriver, 24
DeleteFeature
    OGRLayer, 17
DeleteLayer
    OGRDataSource, 7
Dereference
    OGRDataSource, 10
    OGRLayer, 22

ExecuteSQL
    OGRDataSource, 9

GetDriver
    OGRDataSource, 11
    OGRSFDriverRegistrar, 27
GetDriverCount
    OGRSFDriverRegistrar, 27
GetExtent
    OGRLayer, 18
GetFeature
    OGRLayer, 16
GetFeatureCount
    OGRLayer, 18
GetFIDColumn
    OGRLayer, 21
GetGeometryColumn
    OGRLayer, 22
GetInfo
    OGRLayer, 20
GetLayer
    OGRDataSource, 6
GetLayerByName
    OGRDataSource, 7
GetLayerCount
    OGRDataSource, 6
GetLayerDefn
    OGRLayer, 18
GetName
    OGRDataSource, 6
    OGRSFDriver, 23
GetNextFeature
    OGRLayer, 15
GetRefCount
    OGRDataSource, 11
    OGRLayer, 22
GetRegistrar
    OGRSFDriverRegistrar, 26
GetSpatialFilter
    OGRLayer, 14
GetSpatialRef
    OGRLayer, 18
GetStyleTable
    OGRDataSource, 9
    OGRLayer, 21
GetSummaryRefCount
    OGRDataSource, 11

OGRDataSource, 5
    CreateLayer, 8
    DeleteLayer, 7
    Dereference, 10
    ExecuteSQL, 9
    GetDriver, 11
    GetLayer, 6
    GetLayerByName, 7
    GetLayerCount, 6
    GetName, 6
    GetRefCount, 11
    GetStyleTable, 9
    GetSummaryRefCount, 11
    Reference, 10
    Release, 11
    ReleaseResultSet, 10
    SetDriver, 11
    SetStyleTable, 9
    SetStyleTableDirectly, 9
    SyncToDisk, 10
    TestCapability, 7
OGRLayer, 13
    CreateFeature, 17

CreateField, 20
DeleteFeature, 17
Dereference, 22
GetExtent, 18
GetFeature, 16
GetFeatureCount, 18
GetFIDColumn, 21
GetGeometryColumn, 22
GetInfo, 20
GetLayerDefn, 18
GetNextFeature, 15
GetRefCount, 22
GetSpatialFilter, 14
GetSpatialRef, 18
GetStyleTable, 21
Reference, 22
ResetReading, 15
SetAttributeFilter, 15
SetFeature, 17
SetNextByIndex, 16
SetSpatialFilter, 14
SetSpatialFilterRect, 14
SetStyleTable, 21
SetStyleTableDirectly, 21
SyncToDisk, 20
TestCapability, 19
OGRRegisterAll
ogrsf_frmts.h, 30
ogrsf_frmts.h, 29
OGRRegisterAll, 30
OGRSFDriver, 23
CreateDataSource, 24
DeleteDataSource, 24
GetName, 23
Open, 23
TestCapability, 24
OGRSFDriverRegistrar, 26
GetDriver, 27
GetDriverCount, 27
GetRegistrar, 26
Open, 26
RegisterDriver, 27
Open
OGRSFDriver, 23
OGRSFDriverRegistrar, 26

Reference
OGRDataSource, 10
OGRLayer, 22
RegisterDriver
OGRSFDriverRegistrar, 27
Release
OGRDataSource, 11
ReleaseResultSet
OGRDataSource, 10
ResetReading
OGRLayer, 15

SetAttributeFilter
OGRLayer, 15
SetDriver
OGRDataSource, 11
SetFeature
OGRLayer, 17
SetNextByIndex
OGRLayer, 16
SetSpatialFilter
OGRLayer, 14
SetSpatialFilterRect
OGRLayer, 14
SetStyleTable
OGRDataSource, 9
OGRLayer, 21
SetStyleTableDirectly
OGRDataSource, 9
OGRLayer, 21
SyncToDisk
OGRDataSource, 10
OGRLayer, 20

TestCapability
OGRDataSource, 7
OGRLayer, 19
OGRSFDriver, 24