MapServer Reference Guide for Vector Data Access

By Jeff McKenna and Tyler Mitchell
February, 2005

This document was presented as part of a workshop given at the Open Source GIS / MapServer Users Meeting in Ottawa, Canada 2004.

This is a comprehensive reference guide to using different vector data formats with MapServer. This guide is also available on the MapServer website:

http://mapserver.gis.umn.edu/doc.html

For comments, additions or corrections, please contact one of the authors at:

Tyler Mitchell - tylermitchell@shaw.ca
Jeff McKenna - jmckenna@dmsolutions.ca

Vector Data

What is vector data? This quote from is a good description of what vector data is:

> Vector: "An abstraction of the real world where positional data is represented in the form of coordinates. In vector data, the basic units of spatial information are points, lines and polygons. Each of these units is composed simply as a series of one or more coordinate points. For example, a line is a collection of related points, and a polygon is a collection of related lines. Vector images are defined mathematically as a series of points joined by lines. Vector-based drawings are resolution independent. This means that they appear at the maximum resolution of the output device, such as a printer or monitor. Each object is self-contained, with properties such as color, shape, outline, size, and position on the screen."
>
> From: *http://coris.noaa.gov/glossary/glossary_l_z.html#v*

MapServer can access vector file formats and database connections. It can also access raster or image data. This is a summary of the vector and database formats with particular focus on how to use them in a MapServer map file and access them using command line tools like `ogrinfo`.

MapServer and vector data access

MapServer has two main methods for accessing data:

Through built-in, format-specific, data access capabilities.
: The most basic form of data access uses the built-in capabilities that were *linked into* MapServer when it was compiled. These capabilities are limited to only a few types of vector data, such as ESRI Shapefile, PostGIS, Oracle Spatial and ArcSDE. The default, built-in, format for MapServer is the ESRI Shapefile format.

Using the capabilities of third-party data access libraries.
: One of the most powerful features of MapServer is the ability to use data formats through a pseudo *plug-in* environment. The most significant third party library being used is GDAL/OGR. This includes raster (GDAL) and vector (OGR) data.

Using OGR

OGR is used behind the scenes by MapServer when requested. MapServer does not require OGR in order to run. Some users may never need the additional capabilities OGR offers, but many users find them absolutely necessary. Because MapServer is capable of accessing data via OGR, there is less need to program specific types of data format support directly into MapServer. Instead, further additions can be made to OGR and then be used by MapServer. In essence, the background libraries allow MapServer to bring the data into an internal, memory-based format that MapServer can use. For the most part, using OGR-related formats is seamless and intuitive.

Data Format Types

There are three types of data mapping and GIS data formats. Each type is handled differently. Below are the types and some example formats.

> **File-based** - Shapefiles, Microstation Design Files (DGN), GeoTIFF images
>
> **Directory-based** - ESRI ArcInfo Coverages, US Census TIGER
>
> **Database connections** - PostGIS, ESRI ArcSDE, MySQL

Each type of data is made up of a data source and (one or more) layers. These two definitions apply to MapServer and OGR.

> **Data Source** - a group of layers stored in a common repository. This may be a file that handles several layers within it, or a folder that has several files.
>
> **Layer** - a sub-set of a data source often containing information in one type of vector format (point, line, polygon).

File-based Data

File-based data consists of one or more files stored in any arbitrary folder. In many cases a single file is used (e.g. DGN) but ESRI Shapefiles, for example, consist of at least 3 files each with a different filename extension: `SHP`, `DBF`, `SHX`. In this case all 3 files are required because they each perform a different task internally.

Filenames usually serve as the data source name and contain layers that may or may not be obvious from the filename. In Shapefiles, for example, there is one data source per shapefile and one layer which has the same name as that of the file.

Directory-based Data

Directory-based data consists of one or more files stored in a particular way within a parent folder. In some cases (e.g. Coverages) they may also require additional folders in other locations in the file tree in order to be accessed. The directory itself may be the data source. Different files within the directory often represent the layers of data available.

For example, ESRI ArcInfo Coverages consist of more than one file with an `ADF` file extension, within a folder. The `PAL.ADF` file represents the Polygon data. `ARC.ADF` holds the arc or line string data. The folder holds the data source and each `ADF` file is a layer.

Database Connections

Database Connections are very similar to file and directory-based structures in one respect: they provide geographic coordinate data for MapServer to interpret. That may be oversimplifying what is happening inside MapServer, but in essence all you need is access to the coordinates making up the vector datasets.

Database connections provide a stream of coordinate data that is temporarily stored (e.g. in memory) and read by MapServer to create the map. Other attribute or tabular data may also be required, but the focus of this guide is coordinate data.

One important distinction between databases must be made. The databases discuss here are *spatial* databases, those which can hold geographic data in its own data type. This is opposed to strictly *tabular* databases which cannot hold geographic coordinates in the same way. It is possible to store some *very simple* coordinate data in regular tables, but for anything but the most simple use a spatial database is required. There are spatial extensions to many databases (open source and commercial). One of the most robust is the PostGIS extension to the PostgreSQL database. This database not only allows the storage of geographic data, but also allows the manipulation of that data using SQL commands. The other open source database with spatial capabilities is MySQL.

Connections to databases usually consist of the following pieces of connection information:

Host
: Directions to the server or computer hosting the database

Database name
: The name of the database you wish to access that is running on the host.

User name / passwords
: Access privileges are usually restricted by user

> Some databases (e.g. Oracle) use a name service identifier that includes both the host and database names.

Access to specific pieces of coordinate data usually require:

Table/View name
: The name of the table or view holding the coordinate data

Geographic column name
: Where the geometry or coordinates are stored

Data Format Guide

The rest of this document is the data format guide. This guide is structured to show the fundamentals of each MapServer supported data format. Each section discusses one format, ranging from one to several pages in length. The sections typically start with a summary of the most important information about the format, followed by examples of file listings, connection methods, `ogrinfo` usage and MapServer map file syntax examples.

Each section has been designed to stand alone, so you may notice that certain warnings and comments are repeated or redundant. This is intentional. Each format is presented in rough order of popular use, based on a survey of the MapServer community.

The following formats are included:

ESRI Shapefiles (SHP)
PostGIS / PostgreSQL Database
MapInfo Files (TAB/MID/MIF)
Oracle Spatial Database
Web Feature Service (WFS)
Geography Markup Language (GML)
Virtual Spatial Data (ODBC/OVF)
TIGER Files
ESRI Binary Coverages (ADF)
ESRI ArcSDE (SDE)
Microstation Design Files (DGN)
IHO S-57 Files
Spatial Data Transfer Standard (SDTS)
Inline MapServer Features
National Transfer Format (NTF)

MySQL spatial database is not covered in this guide at this time due to lack of familiarity by the authors. Future contributions by MySQL users are welcome in this guide.

ESRI Shapefiles (SHP)

Also known as ESRI ArcView Shapefiles or ESRI Shapefiles. ESRI is the company that introduced this format. ArcView was the first product to use shapefiles.

File listing

Shapefiles are made up of a minimum of three similarly named files, with different suffixes:

```
Countries_area.dbf
Countries_area.shp
Countries_area.shx
```

Data Access / Connection Method

Shapefile access is built directly into MapServer. It is also available through OGR, but direct access without OGR is recommended and discussed here.

The path to the shapefile is required. No file extension should be specified.

Shapefiles only hold one *layer* of data, therefore no distinction needs to be made.

OGRINFO Examples

The directory can serve as a data source.

Each shapefile in a directory serves as a layer.

A shapefile can also be a data source. In this case the layer has the same prefix as the shapefile.

Using ogrinfo on a directory with multiple shapefiles

```
> ogrinfo /data/shapefiles/
INFO: Open of `/data/shapefiles/'
using driver `ESRI Shapefile' successful.
1: wpg_h2o (Line String)
2: wpg_roads (Line String)
3: wpg_roads_dis (Line String)
4: wpgrestaurants (Point)
```

Using ogrinfo on a single shapefile

```
> ogrinfo /data/shapefiles/Countries_area.shp
Had to open data source read-only.
INFO: Open of `Countries_area.shp'
```

```
using driver `ESRI Shapefile' successful.
1: Countries_area (Polygon)
```

Using ogrinfo to examine the structure of the file/layer

```
> ogrinfo -summary /data/shapefiles/Countries_area.shp
Countries_area
Had to open data source read-only.
INFO: Open of `Countries_area.shp'
using driver `ESRI Shapefile' successful.

Layer name: Countries_area
Geometry: Polygon
Feature Count: 27458
Extent: (-180.000000, -90.000000) - (180.000000, 83.627419)
Layer SRS WKT:
(unknown)
FAC_ID: Integer (5.0)
TILE: Integer (3.0)
ARCLIST: String (254.0)
NAM: String (77.0)
PERIMETER: Real (22.17)
POLYGONCOU: Integer (6.0)
NA2DESC: String (45.0)
```

Map File Example

```
LAYER
  NAME my_shapefile
  TYPE POLYGON
  DATA countries_area
  STATUS OFF
  CLASS
    NAME "Countries"
    OUTLINECOLOR 0 0 0
  END
END
```

PostGIS / PostgreSQL Database

PostGIS is Refraction Research's spatial extension to the PostgreSQL enterprise database.

PostGIS Support

PostGIS is supported directly by MapServer and must be compiled into MapServer to work.

In most cases, PostgreSQL and PostGIS libraries (`.dll` or `.so`) must be present in the system's path environment for functionality to be present. This includes the `libpq` and `libpostgis` libraries.

Map File Example

Specify `CONNECTIONTYPE POSTGIS`

Define `CONNECTION` as:
"host=***yourhostname*** dbname=***yourdatabasename*** user=***yourdbusername*** password=***yourdbpassword*** port=***yourpgport***"

`CONNECTION` parameters can be in any order. Most are optional. `dbname` is required. `host` defaults to `localhost`, `port` defaults to `5432` - the standard port for PostgreSQL.

Define `DATA` as: "`geometrycolumn from yourtablename`". MapServer had a bug related to the keyword `from`. Specify it in lower case to avoid problems. `Geometrycolumn` could be `the_geom` if the `shp2pgsql` utility is used to load data, or `wkb_geometry` if `ogr2ogr` was used.

For example:

```
LAYER
  NAME pg_test
  TYPE POLYGON
  CONNECTIONTYPE POSTGIS
  CONNECTION "host=mapserver.com dbname=gmap user=julio"
  DATA "wkb_geometry FROM province"
  CLASS
     ...
  END
END
```

For more info about PostGIS and MapServer see:

PostGIS docs:
http://postgis.refractions.net/docs/

MapInfo Files (TAB/MID/MIF)

MapInfo files are also known as TAB or MID/MIF files.

File listing

The following files are also associated with .TAB files: .DAT, .ID, .MAP. An example is:

```
border.DAT border.ID border.MAP border.TAB
```

The term MID/MIF refers to files with .MID and .MIF extension.

Data Access / Connection Method

TAB and MID/MIF access is available in MapServer through OGR.

The `CONNECTIONTYPE OGR` parameter must be used.

The path to the (`*.tab` or `*.mif`) file is required, and the file extension is needed.

The path may be relative to the `SHAPEPATH`

MapInfo files already contain *styling* information. This styling information can be used optionally by specifying the `STYLEITEM "AUTO"` parameter in the LAYER object of the map file.

If you use `STYLEITEM "AUTO"` you must have an empty class in the layer.

OGRINFO Examples

Using ogrinfo on a single TAB file

```
> ogrinfo elev5_poly.TAB
Had to open data source read-only.
INFO: Open of `elev5_poly.TAB'
using driver `MapInfo File' successful.
1: elev5_poly (Polygon)
```

Using ogrinfo to examine the structure of the file/layer

```
> ogrinfo elev5_poly.TAB elev5_poly
Had to open data source read-only.
INFO: Open of `elev5_poly.TAB'
using driver `MapInfo File' successful.


Layer name: elev5_poly
```

```
Geometry: Polygon
Feature Count: 2236
Extent: (-141.000000, 60.000000) - (-124.403310, 69.300251)
Layer SRS WKT:
GEOGCS["unnamed",
    DATUM["MIF 0",
        SPHEROID["WGS 84 (MAPINFO Datum
0)",6378137.01,298.257223563],
        TOWGS84[0,0,0,0,0,0,0]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]]
AREA: Real (0.0)
PERIMETER: Real (0.0)
ELEV5_: Integer (0.0)
ELEV5_ID: Integer (0.0)
TYPE: Real (4.0)
ELEV5: Real (4.0)
...
```

Map File Example

```
LAYER
  NAME Elevation_Poly_5
  TYPE POLYGON
  STATUS DEFAULT
  CONNECTIONTYPE OGR
  CONNECTION "./hypso/elev5_poly.TAB"
  STYLEITEM "AUTO"
  CLASS
    NAME "Elevation Poly 5"
  END
END # Layer
```

Oracle Spatial Database

MapServer can support Oracle Spatial through OGR.

OGR must be compiled with Oracle Spatial support and MapServer must be compiled to use OGR.

MapServer also supports Oracle Spatial natively.

For more information about Oracle Spatial and MapServer see the MapServer documentation and reference pages.
http://mapserver.gis.umn.edu/doc.html

Map file example using OGR Support

```
LAYER
  ...
  CONNECTION "OCI:user/pwd@service"
  CONNECTIONTYPE OGR
  DATA "Tablename"
  ...
END
```

Example:

```
LAYER
   ...
   NAME "Ottawa"
   CONNECTIONTYPE OGR
   CONNECTION "OCI:jeff/blah@ora_cities"
   DATA "CITIES"
   TYPE POINT
   ...
END
```

Map file example Using Native Support

```
LAYER
   ...
   CONNECTIONTYPE oraclespatial
   CONNECTION "user/pwd@service"
   DATA "GEOMETRY FROM tablename"
   ...
END
```

Web Feature Service (WFS)

WFS is an Open Geospatial Consortium (OGC) specification. For more information about the format itself, see: *http://www.opengeospatial.org/*

WFS allows a client to retrieve geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services. GML is built on the standard web language XML.

WFS differs from the popular Web Map Service (WMS) specification in that WFS returns a subset of the data in valid GML format, not just a graphic image of data.

Capabilities

Requesting the capabilities using the `GetCapabilities` request to a WFS server returns an XML document showing what layers and projections are available, etc.

Example of a WFS GetCapabilities URL

```
http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap
?VERSION=1.0.0
&SERVICE=wfs
&REQUEST=GetCapabilities
```

Example of the Resulting XML from GetCapabilties

```
...
<FeatureTypeList>
  <Operations>
    <Query/>
  </Operations>
    <FeatureType>
        <Name>park</Name>
        <Title>Parks</Title>
        <SRS>EPSG:42304</SRS>
        <LatLongBoundingBox minx="-173.433" miny="41.4271"
maxx="-13.0481" maxy="83.7466" />
    </FeatureType>
    <FeatureType>
        <Name>road</Name>
        <Title>Roads</Title>
        <SRS>EPSG:42304</SRS>
        <LatLongBoundingBox minx="-148.059" miny="35.882"
maxx="-33.7745" maxy="72.5503" />
    </FeatureType>
    <FeatureType>
        <Name>popplace</Name>
        <Title>Cities</Title>
        <SRS>EPSG:42304</SRS>
        <LatLongBoundingBox minx="-172.301" miny="36.3541"
maxx="-12.9698" maxy="83.4832" />
```

```
    </FeatureType>
</FeatureTypeList>
...
```

Data Access / Connection Method

WFS access is a core MapServer feature.

MapServer currently supports WFS version 1.0.0

The `CONNECTIONTYPE WFS` parameter must be used.

WFS layers can be requested through a layer in a map file, or you can request the GML directly through the browser with a `GetFeature` request. You can specify a specific layer with the `TypeName` request. In a map file the name/value pairs should be put into a `METADATA` object.

You can limit the number of features returned in the GML by using the `MaxFeatures` option (e.g. `&MAXFEATURES=100`).

Example of a WFS Request Directly Through the Browser

The following URL requests the GML for the layer `road`. (see the `GetCapabilities` above for the possible layers available on this test server) . The URL is all one line, broken up here for readability.

```
http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap
?VERSION=1.0.0
&SERVICE=wfs
&REQUEST=getfeature&TYPENAME=road
```

Map File Example

```
LAYER
  NAME "wfs_gmap_roads"
  STATUS DEFAULT
  TYPE LINE
  CONNECTIONTYPE WFS
  CONNECTION "http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap?
  METADATA
    "wfs_version" "1.0.0"
    "wfs_srs" "EPSG:42304"
    "wfs_typename" "road"
    "wfs_request_method" "GET"
    "wfs_service" "WFS"
  END
  CLASS
    NAME "roads"
    COLOR 0 0 0
  END
END  # layer
```

Geography Markup Language Files (GML)

Also known as Geographic Markup Language and GML/XML.

GML is a text-based, XML format that can represent vector and attribute data.

This is an Open Geospatial Consortium specification for data interchange.

File listing

GML files are usually a single text file with a GML filename extension. Some may use XML as the filename extension:

```
coal_dep.gml
```

XML schema documents often accompany GML files that have been translated from some other format (e.g. using the `ogr2ogr` utility).

Example of text in a GML file

GML uses sets of nested *tags* to define attributes and geometry coordinates.

```
<gml:featureMember>
  <Coal_Deposits fid="1">
    <UNKNOWN>0.000</UNKNOWN>
    <NA>0.000</NA>
    <ID>2</ID>
    <ID2>2</ID2>
    <MARK>7</MARK>
    <COALKEY>110</COALKEY>
    <COALKEY2>110</COALKEY2>
    <ogr:geometryProperty>
      <gml:Point>
        <gml:coordinates>78.531,50.694</gml:coordinates>
      </gml:Point>
    </ogr:geometryProperty>
  </Coal_Deposits>
</gml:featureMember>
```

Data Access / Connection Method

GML access is available in MapServer through OGR.

The `CONNECTIONTYPE OGR` parameter must be used.

The path to the GML file is required, including file extension.

There can be multiple layers in a GML file, including multiple feature types.

OGRINFO Examples

Using ogrinfo on a single GML file

```
> ogrinfo /data/gml/coal_dep.gml
Had to open data source read-only.
INFO: Open of `coal_dep.gml'
using driver `GML' successful.
1: Coal_Deposits
```

Using ogrinfo to examine the structure of one layer

```
> ogrinfo -summary /data/gml/coal_dep.gml Coal_Deposits
Had to open data source read-only.
INFO: Open of `coal_dep.gml'
using driver `GML' successful.

Layer name: Coal_Deposits
Geometry: Unknown (any)
Feature Count: 266
Extent: (23.293650, 37.986340) - (179.272550, 80.969670)
Layer SRS WKT:
(unknown)
UNKNOWN: Real (0.0)
NA: Real (0.0)
ID: Integer (0.0)
ID2: Integer (0.0)
MARK: Integer (0.0)
COALKEY: Integer (0.0)
COALKEY2: Integer (0.0)
LONG: Real (0.0)
LAT: Real (0.0)
```

Map File Example

```
LAYER
  NAME coal_deposits
  TYPE POINT
  STATUS DEFAULT
  CONNECTIONTYPE OGR
  CONNECTION "gml/coal_dep.gml"
  CLASS
    COLOR 0 0 0
    SYMBOL 'circle'
    SIZE 6
  END
END
```

Virtual Spatial Data (ODBC/OVF)

This is an OGR extension to MapServer. It allows you to connect to databases that do not explicitly hold spatial data, as well as flat text files. Your data must have an X and a Y column, and the data may be accessed through an ODBC connection or a direct pointer to a text file.

Types of Databases

The VirtualSpatialData OGR extension has been tested with the following databases and should, in theory, support all ODBC data sources.

- Oracle
- MySQL
- SQL Server
- Access
- PostgreSQL

Types of Flat Files

Comma, tab or custom delimited text/flat files work with VirtualSpatialData.

Create the Datasource Name (DSN)

Specific notes about creating a DSN on Windows and Linux can be found by searching the MapServer reference documents site:

http://mapserver.gis.umn.edu/doc

On some Windows systems you *must* create a *SYSTEM* DSN.

Test your Connection

Test your connection with `ogrinfo`. The syntax for this command is:

```
> ogrinfo ODBC:user/pass@DSN table
```

OGRINFO Examples

Accessing a comma separated text file through ODBC.

The following is a snippet of the flat text file `coal_dep.txt` containing lat/long points:

```
unknown,na,id,id2,mark,coalkey,coalkey2,long,lat
0.000,0.000,1,1,7,87,87,76.90238,51.07161
0.000,0.000,2,2,7,110,110,78.53851,50.69403
0.000,0.000,3,3,3,112,112,83.22586,71.24420
0.000,0.000,4,4,6,114,114,80.79896,73.41175
```

If the DSN name is `Data_txt`, the `ogrinfo` command to see a list of applicable files in the directory is:

```
> ogrinfo ODBC:jeff/test@Data_txt
INFO: Open of `ODBC:jeff/test@Data_txt'
using driver `ODBC' successful.
1: coal_dep.csv
2: coal_dep.txt
3: coal_dep_nf.txt
4: coal_dep_trim.txt
5: Copy of coal_dep.txt
6: deposit.csv
7: maruia.asc
8: oahuGISbathy.csv
9: oahuGISbathy.txt
10: on_pts.txt
11: on_pts_utm.txt
12: test.txt
13: utm_test.txt
```

Username and password may be optional, so the following may also be valid:

```
> ogrinfo ODBC:@Data_txt
```

Therefore, the command to see more information about one of the specific layers is:

```
> ogrinfo ODBC:@Data_txt coal_dep.txt
INFO: Open of `ODBC:@Data_txt'
using driver `ODBC' successful.

Layer name: coal_dep.txt
Geometry: Unknown (any)
Feature Count: 266
Layer SRS WKT:
(unknown)
UNKNOWN: String (255.0)
NA: String (255.0)
ID: String (255.0)
ID2: String (255.0)
MARK: String (255.0)
COALKEY: String (255.0)
COALKEY2: String (255.0)
LONG: String (255.0)
LAT: String (255.0)
OGRFeature(coal_dep.txt):0
  UNKNOWN (String) = 0.000
  ....
```

Create a Virtual Data File

This is a file with an `ovf` extension and looks like the following:

```
<OGRVRTDataSource>
    <OGRVRTLayer name="mylayer">
        <SrcDataSource>ODBC:user/pass@DSN</SrcDataSource>
    <SrcLayer>tablename</SrcLayer>
   <GeometryType>wkbPoint</GeometryType>
        <LayerSRS>WGS84</LayerSRS>
   <GeometryField encoding="PointFromColumns" x="x" y="y"/>
    </OGRVRTLayer>
</OGRVRTDataSource>
```

More information on `ovf` files can be found at:
http://www.remotesensing.org/gdal/ogr/drv_vrt.html

Example ovf file for coal_dep.txt

```
<OGRVRTDataSource>
  <OGRVRTLayer  name="coal">
     <SrcDataSource>ODBC:Data_txt</SrcDataSource>
     <SrcLayer>coal_dep.txt</SrcLayer>
     <GeometryField encoding="PointFromColumns" x="Long"
y="Lat"/>
     <GeometryType>wkbPoint</GeometryType>
  </OGRVRTLayer>
</OGRVRTDataSource>
```

Map File Example

Using an `ovf` file your layer may look like:

```
LAYER
 CONNECTION "coal.ovf"
 CONNECTIONTYPE OGR
 DATA "coal-test"
   METADATA
     "wms_srs"   "4326"
     "wms_title"   "coal-test"
   END
 NAME "coal-test"
 SIZEUNITS PIXELS
 STATUS ON
 TOLERANCE 0
 TOLERANCEUNITS PIXELS
 TYPE POINT
 UNITS METERS
 CLASS
   STYLE
       COLOR 255 0 0
     MAXSIZE 100
     MINSIZE 1
     SIZE 6
     SYMBOL "star"
   END
 END
END
```

Or you may specify the `ovf` contents *inline* such as:

```
LAYER
  CONNECTION "<OGRVRTDataSource>
```

```
<OGRVRTLayer  name='coal-test'>
<SrcDataSource>ODBC:@Data_txt</SrcDataSource>
<SrcLayer>coal_dep.txt</SrcLayer>
<GeometryField encoding='PointFromColumns' x='Long' y='Lat'/>
<GeometryType>wkbPoint</GeometryType>
</OGRVRTLayer>
</OGRVRTDataSource>"
    CONNECTIONTYPE OGR
    DATA "coal-test"
      METADATA
        "wms_srs"   "4326"
        "wms_title"   "coal-test"
      END
    NAME "coal-test"
    SIZEUNITS PIXELS
    STATUS ON
    TOLERANCE 0
    TOLERANCEUNITS PIXELS
    TYPE POINT
    UNITS METERS
    CLASS
      STYLE
          COLOR 255 0 0
        MAXSIZE 100
        MINSIZE 1
        SIZE 6
        SYMBOL "star"
      END
    END
  END
```

TIGER/Line Files

TIGER/Line files are created by the US Census Bureau and cover the entire US. They are often referred simply as TIGER files. For more information see:
http://www.census.gov/geo/www/tiger/.

File listing

TIGER/Line files are text files and directory-based data sources. For example, one county folder `TGR06059` contains several associated files:

```
TGR06059.RT1 TGR06059.RT2 TGR06059.RT4 TGR06059.RT5
TGR06059.RT6 TGR06059.RT7 TGR06059.RT8 TGR06059.RTA
TGR06059.RTC TGR06059.RTH TGR06059.RTI TGR06059.RTP
TGR06059.RTR TGR06059.RTS TGR06059.RTT TGR06059.RTZ
```

Data Access / Connection Method

TIGER/Line access occurs through OGR

The *full* path to the directory containing the associated files is required in the `CONNECTION` string. The layer number is added to the `CONNECTION` string, after the path, separated by a comma. e.g.:
`CONNECTION "/tiger/data,0"`

The layer number in the map file is actually the `ogrinfo` layer number, minus one.

OGRINFO Examples

Using ogrinfo on a TIGER directory to retrieve layer numbers

```
> ogrinfo TGR06059 (NOTE that this is a directory)
ERROR 4: Tiger Driver doesn't support update.
Had to open data source read-only.
INFO: Open of `TGR06059'
using driver `TIGER' successful.
1: CompleteChain (Line String)
2: AltName (None)
3: FeatureIds (None)
4: ZipCodes (None)
5: Landmarks (Point)
6: AreaLandmarks (None)
7: Polygon (None)
8: PolygonCorrections (None)
9: EntityNames (Point)
10: PolygonEconomic (None)
11: IDHistory (None)
12: PolyChainLink (None)
```

```
13: PIP (Point)
14: TLIDRange (None)
15: ZeroCellID (None)
16: OverUnder (None)
17: ZipPlus4 (None)
```

For the `CompleteChain` Line layer, the `ogrinfo` layer number is `1`. However, when referring to this layer in a map file `CONNECTION` the layer number will be one less, `0`.

Using ogrinfo to examine the structure of the TIGER layer CompleteChain

```
> ogrinfo TGR06059 CompleteChain
ERROR 4: Tiger Driver doesn't support update.
Had to open data source read-only.
INFO: Open of `TGR06059'
using driver `TIGER' successful.

Layer name: CompleteChain
Geometry: Line String
Feature Count: 123700
Extent: (-118.125898, 33.333992) - (-117.412987, 33.947512)
Layer SRS WKT:
GEOGCS["NAD83",
    DATUM["North_American_Datum_1983",
        SPHEROID["GRS 1980",6378137,298.257222101]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]]
MODULE: String (8.0)
TLID: Integer (10.0)
SIDE1: Integer (1.0)
SOURCE: String (1.0)
FEDIRP: String (2.0)
FENAME: String (30.0)
FETYPE: String (4.0)
FEDIRS: String (2.0)
CFCC: String (3.0)
FRADDL: String (11.0)
TOADDL: String (11.0)
FRADDR: String (11.0)
TOADDR: String (11.0)
FRIADDL: String (1.0)
TOIADDL: String (1.0)
FRIADDR: String (1.0)
TOIADDR: String (1.0)
ZIPL: Integer (5.0)
```

Map File Example

```
LAYER
  NAME Complete_Chain
  TYPE LINE
  STATUS DEFAULT
  CONNECTIONTYPE OGR
  CONNECTION "/path/to/data/tiger/TGR06059,0"
  CLASS
    COLOR 153 102 0
  END
END # Layer
```

ESRI ArcInfo Coverage Files

ESRI ArcInfo Coverage Files are also known as simply as Coverages and, less commonly as ADF files.

File listing

Coverages are made up of a set of files within a folder. The folder itself is the coverage name. The files roughly represent different layers, usually representing different types of topology or feature types.

```
> ls /data/coverage/brazil
aat.adf  arc.adf  arx.adf  bnd.adf  lab.adf  prj.adf  tic.adf
tol.adf
```

A folder with the name `INFO` is also part of the coverage. It sits at the same hierarchical level as the coverage folder itself. Therefore, to copy a coverage (using regular file system tools) the coverage folder and the `INFO` folder must both be copied. The `INFO` folder holds some catalogue information about the coverage.

```
> ls /data/coverage/info
arc0000.dat  arc0001.dat  arc0002.dat  arc.dir
arc0000.nit  arc0001.nit  arc0002.nit
```

Data Access / Connection Method

`CONNECTIONTYPE OGR` must be used. The ability to use coverages is not built into MapServer.

The path to the coverage folder name is required.

The layer number is used to specify what type of features to draw (as per the layer number from `ogrinfo`, but minus one)

OGRINFO Examples

The directory is the data source.

Layers are found within the directory.

Using ogrinfo on a coverage directory

```
> ogrinfo /data/coverage/brazil
INFO: Open of `brazil'
using driver `AVCBin' successful.
1: ARC (Line String)
2: CNT (Point)
3: LAB (Point)
4: PAL (Polygon)
```

Using ogrinfo to examine the structure of a layer

```
> ogrinfo -summary /data/coverage/brazil PAL
Had to open data source read-only.
INFO: Open of `brazil'
using driver `AVCBin' successful.

Layer name: PAL
Geometry: Polygon
Feature Count: 1
Extent: (1272793.274958, 795381.617050) - (1287078.382785,
807302.747284)
Layer SRS WKT:
(unknown)
ArcIds: IntegerList (0.0)
AREA: Real (18.5)
PERIMETER: Real (18.5)
F_OPER#: Integer (5.0)
F_OPER-ID: Integer (5.0)
OPER: String (2.0)
FCODE: String (10.0)
```

Map File Example

```
LAYER
  NAME Brazil_bounds
  TYPE POLYGON
  STATUS DEFAULT
  CONNECTIONTYPE OGR
  CONNECTION "/data/coverage/brazil, 3"
  CLASS
    NAME "Brazil Admin Areas"
    OUTLINECOLOR 153 102 0
    SIZE 2
  END
END
```

ArcSDE Database (SDE)

ArcSDE is ESRI's spatial plugin for SQL Server, Oracle and DB2 databases. It is also known as Spatial Database Engine, but most commonly as simply SDE.

SDE Support

SDE is supported directly by MapServer.

MapServer 4.4 supports: SDE 9, 8 and SDE for Coverages on Linux, Windows, and Solaris platforms.

Recent developments in MapServer (v4.4) have added support for SDE raster layers. Using raster layers is beyond the scope of this guide.

Connecting to SDE

Install the SDE client libraries from the SDE CDs

Compile MapServer with SDE support

Define the layer in the map file:

Map File Example

Specify `CONNECTIONTYPE SDE`

Define the `CONNECTION` as: hostname, instancename, databasename, username,password

Define the `DATA` as: tablename, geometrycolumn

For example:

```
LAYER
  NAME         test
  TYPE         POLYGON
  CONNECTION "sde.dms.ca,port:5151,sde,user,password"
  CONNECTIONTYPE SDE
  DATA "NTDB.WATER,SHAPE"
  CLASS
     ...
  END
END
```

For more about SDE and MapServer see the MapServer documentation page or search the reference documentation:

http://mapserver.gis.umn.edu/doc

Microstation Design Files (DGN)

Also known as DGN files.

File listing

Data are encapsulated in a single file, usually with the suffix `.dgn`.

```
0824t.dgn
```

Data Access / Connection Method

Access is available in MapServer through OGR.

The `CONNECTIONTYPE OGR` parameter must be used.

The path to the `dgn` file is required, file extension is needed.

All types of features in a DGN file are held in one "layer" of data. The layer is called `elements` and is the first and only layer.

The type of feature to be read from the DGN depends on the `TYPE` parameter in the map file.

DGN files typically contain `POINT`, `LINE`, `POLYGON` and `ANNOTATION` feature types.

DGN files contain "styling" information - how to colour and present the data. This is used, optionally, by specifying the `STYLEITEM "AUTO"` parameter.

> **NOTE**: DGN files typically use white as a colour for their features and therefore are not visible on maps with white backgrounds.

OGRINFO Examples

Using ogrinfo on a single DGN file

Note that no geometry/feature type for the layer is identified because it can be multiple types.

```
> ogrinfo /data/dgn/0824t.dgn
Had to open data source read-only.
INFO: Open of `0842t.dgn'
using driver `DGN' successful.
1: elements
```

Using ogrinfo to examine the structure of the file/layer

DGN files are not really GIS data files. They evolved from drafting formats used by computer aided drafting/design (CADD) programs.

They carry a few key attributes which are usually consistent across all DGN files. Most of the attributes relate to graphical styling of features for map presentation, such as `ColorIndex`, `Style`, etc.

Spatial reference system information is not always encoded into DGN files. This can be a major problem when trying to adequately reference the DGN data in another mapping program.

Measurement units can be a problem. In some cases the features could be located in kilometres or feet even though it is not obvious from the output of `ogrinfo`. Sometimes the only way to identify or correct a problem with units is to open the file in Microstation software.

```
> ogrinfo -summary /data/dgn/0824t.dgn elements
INFO: Open of `0824t.dgn'
using driver `DGN' successful.

Layer name: elements
Geometry: Unknown (any)
Feature Count: 22685
Extent: (-513183.050000, 150292.930000) - (-224583.220000,
407463.360000)
Layer SRS WKT:
(unknown)
Type: Integer (2.0)
Level: Integer (2.0)
GraphicGroup: Integer (4.0)
ColorIndex: Integer (3.0)
Weight: Integer (2.0)
Style: Integer (1.0)
EntityNum: Integer (8.0)
MSLink: Integer (10.0)
Text: String (0.0)
```

Map File Example

```
LAYER
  NAME dgn
  TYPE LINE
  STATUS DEFAULT
  CONNECTIONTYPE OGR
  CONNECTION "dgn/0824t.dgn,0"
  STYLEITEM "AUTO"
  CLASS
  END
END # Layer
```

IHO S-57 Files

Also known as S57. The IHO S-57 format is a vector interchange format used for maritime charts.

Developed by the International Hydrographic Organisation (IHO). For more information about the IHO see:

http://www.iho.shom.fr/

File listing

Individual S57 data files have an extension of `*.000`. e.g. `US1BS02M.000`

Data Access / Connection Method

S57 access in MapServer occurs through OGR, `CONNECTIONTYPE OGR` must be used.

Specify a full path or a relative path from the `SHAPEPATH` to the `.000` file for the `CONNECTION`

The `CONNECTION` must also include a layer number (as per the layer number from `ogrinfo`, but minus one).

Special Notes

The underlying OGR code requires two files from your GDAL/OGR installation when reading S57 data in MapServer : `s57objectclasses.csv` and `s57attributes.csv`. These files can be found in the `/GDAL/data/` folder. If you receive an error in MapServer such as:

```
msDrawMap(): Image handling error. Failed to draw layer named
's57'. msOGRFileOpen(): OGR error. GetLayer( 9) failed for OGR
connection
```

you may have to point MapServer to these files using the `CONFIG` parameter in the main section of your map file:

```
CONFIG GDAL_DATA "C:\gdal\data"
```

OGRINFO Examples

Using ogrinfo on an S57 file to get the OGR index number

```
> ogrinfo us1bs02m.000
ERROR 4: S57 Driver doesn't support update.
Had to open data source read-only.
```

```
INFO: Open of `us1bs02m.000'
using driver `IHO S-57 (ENC)' successful.
1: ADMARE (Polygon)
2: CBLSUB (Line String)
3: CTNARE
4: COALNE (Line String)
5: DEPARE
6: DEPCNT (Line String)
7: LNDARE
8: LNDELV
9: LNDRGN
10: LNDMRK
11: LIGHTS (Point)

12: OBSTRN
13: RDOSTA (Point)
14: SEAARE
15: SBDARE
16: SLCONS
17: SOUNDG (Multi Point)
18: UWTROC (Point)
19: WATTUR
20: WRECKS
21: M_COVR (Polygon)
22: M_NPUB (Polygon)
23: M_NSYS (Polygon)
24: M_QUAL (Polygon)
25: C_ASSO (None)
```

Using ogrinfo to examine the structure of an S57 layer

```
> ogrinfo us1bs02m.000 DEPARE
ERROR 4: S57 Driver doesn't support update.
Had to open data source read-only.
INFO: Open of `us1bs02m.000'
using driver `IHO S-57 (ENC)' successful.

Layer name: DEPARE
Geometry: Unknown (any)
Feature Count: 297
Extent: (165.666667, 48.500000) - (180.000000, 60.750000)
Layer SRS WKT:
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.257223563]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]]
GRUP: Integer (3.0)
OBJL: Integer (5.0)
RVER: Integer (3.0)
AGEN: Integer (2.0)
FIDN: Integer (10.0)
FIDS: Integer (5.0)
LNAM: String (16.0)
LNAM_REFS: StringList (16.0)
DRVAL1: Real (0.0)
DRVAL2: Real (0.0)
QUASOU: String (0.0)
SOUACC: Real (0.0)
VERDAT: Integer (0.0)
INFORM: String (0.0)
NINFOM: String (0.0)
NTXTDS: String (0.0)
SCAMAX: Integer (0.0)
SCAMIN: Integer (0.0)
TXTDSC: String (0.0)
RECDAT: String (0.0)
```

```
RECIND: String (0.0)
...
```

Map File Example

```
LAYER
  NAME s57
  TYPE POLYGON
  STATUS DEFAULT
  CONNECTIONTYPE OGR
  CONNECTION "./s57/us1bs02m.000, 4"
  CLASS
    COLOR 247 237 219
    OUTLINECOLOR 120 120 120
  END
END # Layer
```

Spatial Data Transfer Standard Files (SDTS)

This is a United States Geological Survey (USGS) format

SDTS has a raster and a vector format. The raster format is *not* supported in MapServer. Only the vector formats are supported, including VTP and DLG files.

File listing

SDTS files are often organized into state-sized pieces. For example, all of the state of Maryland (MD), U.S.A.

Files are also available for multiple types of features including hydrography, transportation and administrative boundaries.

This example uses transportation data, which consists of 35 separate files, each with the suffix `DDF`.

```
MDTRAHDR.DDF  MDTRARRF.DDF  MDTRCATS.DDF
MDTRDQCG.DDF  MDTRFF01.DDF  MDTRLE02.DDF
MDTRNA03.DDF  MDTRNO03.DDF  MDTRSPDM.DDF
MDTRAMTF.DDF  MDTRBFPS.DDF  MDTRCATX.DDF
MDTRDQHL.DDF  MDTRIDEN.DDF  MDTRLE03.DDF
MDTRNE03.DDF  MDTRPC01.DDF  MDTRSTAT.DDF
MDTRARDF.DDF  MDTRBMTA.DDF  MDTRDDSH.DDF
MDTRDQLC.DDF  MDTRIREF.DDF  MDTRNA01.DDF
MDTRNO01.DDF  MDTRPC02.DDF  MDTRXREF.DDF
MDTRARDM.DDF  MDTRCATD.DDF  MDTRDQAA.DDF
MDTRDQPA.DDF  MDTRLE01.DDF  MDTRNA02.DDF
MDTRNO02.DDF  MDTRPC03.DDF
```

Data Access / Connection Method

SDTS access is available in MapServer through OGR.

The `CONNECTIONTYPE OGR` parameter must be used.

The path to the *catalog* file (`????CATD.DDF`) is required, including file extension.

There are multiple layers in the SDTS catalog, some of which are only attributes and have no geometries.

OGRINFO Examples

Using ogrinfo on a catalog file

Note that the first 7 layers do not have geometries.

```
> ogrinfo /data/sdts/MD/MDTRCATD.DDF
```

```
Had to open data source read-only.
INFO: Open of `MDTRCATD.DDF'
using driver `SDTS' successful.
1: ARDF (None)
2: ARRF (None)
3: AMTF (None)
4: ARDM (None)
5: BFPS (None)
6: BMTA (None)

7: AHDR (None)
8: NE03 (Point)
9: NA01 (Point)
10: NA02 (Point)
11: NA03 (Point)
12: NO01 (Point)
13: NO02 (Point)
14: NO03 (Point)
15: LE01 (Line String)
16: LE02 (Line String)
17: LE03 (Line String)
18: PC01 (Polygon)
19: PC02 (Polygon)
20: PC03 (Polygon)
```

Using ogrinfo to examine the structure of the file/layer

```
> ogrinfo -summary /data/sdts/MD/MDTRCATD.DDF LE01
Had to open data source read-only.
INFO: Open of `MDTRCATD.DDF'
using driver `SDTS' successful.

Layer name: LE01
Geometry: Line String
Feature Count: 780
Extent: (-80.000289, 36.999774) - (-74.999711, 40.000225)
Layer SRS WKT:
GEOGCS["NAD27",
    DATUM["North_American_Datum_1927",
        SPHEROID["Clarke 1866",6378206.4,294.978698213901]],
    PRIMEM["Greenwich",0],
    UNIT["degree",0.0174532925199433]]
RCID: Integer (0.0)
SNID: Integer (0.0)
ENID: Integer (0.0)
ENTITY_LABEL: String (7.0)
ARBITRARY_EXT: String (1.0)
RELATION_TO_GROUND: String (1.0)
VERTICAL_RELATION: String (1.0)
OPERATIONAL_STATUS: String (1.0)
ACCESS_RESTRICTION: String (1.0)
OLD_RAILROAD_GRADE: String (1.0)
WITH_RAILROAD: String (1.0)
COVERED: String (1.0)
HISTORICAL: String (1.0)
LIMITED_ACCESS: String (1.0)
PHOTOREVISED: String (1.0)
LANES: Integer (2.0)
ROAD_WIDTH: Integer (3.0)
BEST_ESTIMATE: String (1.0)
ROUTE_NUMBER: String (7.0)
ROUTE_TYPE: String (9.0)
```

Map File Example

```
LAYER
  NAME sdts_maryland
  TYPE LINE
  CONNECTIONTYPE OGR
  CONNECTION "data/sdts/MD/MDTRCATD.DDF,14"
  STATUS DEFAULT
  CLASS
    COLOR 0 0 0
  END
END
```

Inline MapServer Features

Inline features refer to coordinates entered directly into the map file. They are not a file or database format and do not require any `DATA` or `CONNECTION` parameters. Instead they use a `FEATURE` section to define the coordinates.

Inline features can be used to define points, lines and polygons as if taken from an external file.

This requires direct entry of coordinate pairs in the map file using a particular syntax.

Data Access / Connection Method

This is a native MapServer option that doesn't use any external libraries to support it.

Map File Example

Points

Each `FEATURE..END` section defines a feature.

Multiple points can be defined in a `FEATURE` section. If multiple points are defined in the same layer, they will have the same `CLASS` settings, e.g. for colours and styles.

Coordinates are entered in the units set in the layer's projection. In this case it is assuming the map file projection is using decimal degrees.

```
LAYER
  NAME inline_stops
  TYPE POINT
  STATUS DEFAULT
  FEATURE
    POINTS
      72.36 33.82
    END
    TEXT "My House"
  END
  FEATURE
    POINTS
      69.43 35.15
      71.21 37.95
      72.02 38.60
    END
    TEXT "My Stores"
  END
  CLASS
    COLOR 0 0 250
    SYMBOL 'circle'
    SIZE 6
  END
END
```

Lines

Lines are simply a list of points strung together, but the layer must be `TYPE LINE` instead of `TYPE POINT`.

```
LAYER
  NAME inline_track
  TYPE LINE
  STATUS DEFAULT
  MAXSCALE 10000000
  FEATURE
    POINTS
      72.36 33.82
      70.85 34.32
      69.43 35.15
      70.82 36.08
      70.90 37.05
      71.21 37.95
    END
  END
  CLASS
    COLOR 255 10 0
    SYMBOL 'circle'
    SIZE 2
  END
END
```

Polygons

Polygons are the same as the line example, just a list of points.

They require the `TYPE POLYGON` parameter.

Polygons require the final coordinate pair to be the same as the first, making it a *closed polygon*.

National Transfer Format Files (NTF)

NTF files are mostly used by the United Kingdom Ordnance Survey (OS). For more on the Ordnance Survey, see their website at:

http://www.ordnancesurvey.co.uk

File listing

NTF files have an `NTF` extension.

Data Access / Connection Method

NTF access requires OGR.

The path to the NTF file is required in the `CONNECTION` string. It may be relative to the `SHAPEPATH` setting in the map file or the full path.

The `CONNECTION` must also include a layer number (as per the layer number from `ogrinfo`, but minus one).

OGRINFO Examples

Using ogrinfo on an NTF file to retrieve layer numbers

```
> ogrinfo llcontours.ntf
ERROR 4: NTF Driver doesn't support update.
Had to open data source read-only.
INFO: Open of `llcontours.ntf'
using driver `UK .NTF' successful.
1: LANDLINE_POINT (Point)
2: LANDLINE_LINE (Line String)
3: LANDLINE_NAME (Point)
4: FEATURE_CLASSES (None)
```

For the `LANDLINE_LINE` layer, the OGRINFO layer number is `2`, however when referring to this layer in a map file CONNECTION the layer number will be `1`.

Using ogrinfo to examine the structure of an NTF layer

```
> ogrinfo llcontours.ntf LANDLINE_LINE
ERROR 4: NTF Driver doesn't support update.
Had to open data source read-only.
INFO: Open of `llcontours.ntf'
using driver `UK .NTF' successful.

Layer name: LANDLINE_LINE
Geometry: Line String
Feature Count: 491
Extent: (279000.000000, 187000.000000) - (280000.000000,
188000.000000)
Layer SRS WKT:
PROJCS["OSGB 1936 / British National Grid",

    GEOGCS["OSGB 1936",
        DATUM["OSGB_1936",
            SPHEROID["Airy 1830",6377563.396,299.3249646,
                AUTHORITY["EPSG","7001"]],
            AUTHORITY["EPSG","6277"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433],
        AUTHORITY["EPSG","4277"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",49],
    PARAMETER["central_meridian",-2],
    PARAMETER["scale_factor",0.999601272],
    PARAMETER["false_easting",400000],
    PARAMETER["false_northing",-100000],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AUTHORITY["EPSG","27700"]]
LINE_ID: Integer (6.0)
FEAT_CODE: String (4.0)
...
```

Map File Example

```
LAYER
  NAME ntf_uk
  TYPE LINE
  CONNECTIONTYPE OGR
  CONNECTION "./ntf/llcontours.ntf,1"
  STATUS DEFAULT
  CLASS
    NAME "Contours"
    COLOR 0 150 200
  END
END
```