

## ASDevMapView : How to read MapViewer's log file

This page last changed on Sep 12, 2007 by [lj\\_qian](#).

In this page we talk about how to read MapViewer 11g's log file, and how the information contained in the log file might help in debugging and tuning your applications.

### When to look at a log file

The log file is a critical piece of information that can help you in the following situations:

- whenever you saw MapViewer related error messages from your application
- whenever you think a map is taking too long to display, displays wrong results, or no data is displayed at all.
- whenever you think the performance of a displaying a map or certain layers is way too slow
- whenever you think something might be wrong with MapViewer

### Where is the log file

Note that you will find the MapViewer log files in different locations depending on your deployment.

Within a full Application Server installation, these logs are found in the opmn's log file for the OC4J instance that contains MapViewer. For instance, if you deployed MapViewer to the OC4J instance named 'home' in your Application Server, then the log file that contains MapViewer log records will typically be \$OAS\_HOME/opmn/logs/OC4J~home~default\_group~1. Simply replace 'home' with the proper name if MapViewer is deployed to a different OC4J instance.

In a standalone OC4J instance, your best bet is to enable MapViewer's own logging by changing the <logging> element in the mapViewerConfig.xml config file. When MapViewer's own logging is enabled, you always have the option of accessing this log file for all things logged by MapViewer. This is your only viable option if you are running inside a standalone OC4J instance.

### Configuring MapViewer logs

To enable and change MapViewer configuration for logs, such as the current logging level, you need to edit the mapViewerConfig.xml file. This is discussed in the MapViewer User's Guide, in the section where the configuration file is discussed. Typically if you are debugging or tuning your application, you need to set the logging level to "finest". The following is a typical logging element configured to log everything:

```
<logging log_level="finest" log_thread_name="false"
log_time="true">
<log_output name="System.err" />
<log_output name="../log/mapviewer.log" />
</logging>
```

### How the log file is organized

If the logging level is set to finest, MapViewer will provide the most detailed (sometimes overly verbose) view of what's going on inside MapViewer. For instance, the following is a snippet of the logs when MapViewer is first started.

```
2007-08-23 10:56:00.888 NOTIFICATION oms root path: /ul/oc4j/mv10131_qs/oc4j/j2ee/home/
applications/mapviewer/web/
07/08/23 10:56:00 mbean domain: mapviewer
2007-08-23 10:56:00.928 NOTIFICATION OMSConfig mbean registred.
2007-08-23 10:56:00.941 NOTIFICATION MapCacheServer root path: /ul/oc4j/mv10131_qs/oc4j/j2ee/
home/applications/mapviewer/web/
2007-08-23 10:56:00.952 NOTIFICATION MapViewer server version: Ver10131_B070815
2007-08-23 10:56:00.957 NOTIFICATION using default config file: /ul/oc4j/mv10131_qs/oc4j/j2ee/
home/applications/mapviewer/web/WEB-INF/conf/mapViewerConfig.xml
07/08/23 10:56:00 mbean domain: mapviewer
```

2007-08-23 10:56:00.970 NOTIFICATION MCSConfig mbean registred.  
2007-08-23 10:56:00.972 WARNING dcsf:true  
2007-08-23 10:56:00.982 WARNING destroying ALL mapmaker instances.  
2007-08-23 10:56:00.987 WARNING dcsf:true  
2007-08-23 10:56:01.081 NOTIFICATION Setting logging options for MapViewer  
2007-08-23 10:56:01.082 NOTIFICATION setting logging level to finest  
Aug 23, 2007 10:56:01 AM oracle.lbs.mapserver.core.MapperConfig loadMapViewConfig  
FINEST:  
Allowed IPs:  
Excluded IPs:

Aug 23, 2007 10:56:01 AM oracle.sdovis.GlobalVisContext <clinit>  
WARNING: will use 96 as default dpi.  
Aug 23, 2007 10:56:01 AM oracle.lbs.mapserver.core.MapperConfig registerNSDataProviders  
FINEST: Non-Spatial Data Provider registered: defaultNSDP  
07/08/23 10:56:01 Oracle Containers for J2EE 10g (10.1.3.1.0) initialized  
Aug 23, 2007 10:56:02 AM oracle.sdovis.SRS <clinit>  
INFO: Using 96 as default dpi.  
Aug 23, 2007 10:56:02 AM oracle.sdovis.DBRSRSCache loadGeodeticSrids  
FINEST: Number of geodetic srids loaded: 844.  
Aug 23, 2007 10:56:02 AM oracle.lbs.mapserver.core.MapperConfig createMappers  
FINEST: Data source mvdemo is associated with a pl/sql package: web\_user\_info  
Aug 23, 2007 10:56:02 AM oracle.sdovis.CacheMgr2 init  
INFO: Spatial Data Cache opened. Region=SDOVIS\_DATA.  
Aug 23, 2007 10:56:02 AM oracle.sdovis.CacheMgr2 init  
INFO: max\_cache\_size=32 MB.  
Aug 23, 2007 10:56:02 AM oracle.sdovis.CacheMgr2 createSubRegion  
INFO: sub region sdovis\_subreg\_mvdemo\_jdbc:oracle:thin:@stadb32.us.oracle.com:15214:mv created  
in cache.  
Aug 23, 2007 10:56:02 AM oracle.lbs.mapserver.core.MapperPool addMapper  
INFO: added a mapper instance to the pool [data src=mvdemo]  
Aug 23, 2007 10:56:02 AM oracle.lbs.mapserver.core.MapperPool addMapper  
INFO: added a mapper instance to the pool [data src=mvdemo]  
Aug 23, 2007 10:56:02 AM oracle.lbs.mapserver.core.MapperPool addMapper  
INFO: added a mapper instance to the pool [data src=mvdemo]  
Aug 23, 2007 10:56:02 AM oracle.lbs.mapserver.core.MapperConfig loadConfigFile  
INFO: Map Recycling thread started.  
Aug 23, 2007 10:56:02 AM oracle.lbs.mapserver.oms\$ColdStart run  
**INFO: \*\*\* Oracle MapViewer started. \*\*\***  
Aug 23, 2007 10:56:02 AM oracle.lbs.foi.FOIServer init  
**INFO: \*\*\* Oracle Feature of Interest (FOI) Server started. \*\*\***  
Aug 23, 2007 10:56:03 AM oracle.lbs.mapcache.config.CacheInstanceConfig loadCacheStorageDef  
WARNING: Invalid cache root directory:/scrtach/mvdemomaps/. Cache root directory will be set to  
default root directory.  
Aug 23, 2007 10:56:03 AM oracle.sdovis.DBRSRSCache register  
INFO: Loading SRS information from MDSYS.cs\_srs (srid=8307).  
07/08/23 10:56:03 Using 96 as default dpi.  
Aug 23, 2007 10:56:03 AM oracle.sdovis.DBRSRSCache parseAndAdd  
FINER: Registering srs 8307, isGeodetic=true, unit=DECIMAL DEGREE  
Aug 23, 2007 10:56:03 AM oracle.lbs.mapcache.config.CacheInstanceConfig loadCacheStorageDef  
WARNING: Invalid cache root directory:/scrtach/mvdemomaps/. Cache root directory will be set to  
default root directory.  
Aug 23, 2007 10:56:03 AM oracle.sdovis.DBRSRSCache register  
INFO: Loading SRS information from MDSYS.cs\_srs (srid=32775).  
Aug 23, 2007 10:56:03 AM oracle.sdovis.DBRSRSCache parseAndAdd  
FINER: Registering srs 32775, isGeodetic=false, unit=METER  
Aug 23, 2007 10:56:03 AM oracle.lbs.mapcache.config.CacheInstanceConfig loadCacheStorageDef  
WARNING: Invalid cache root directory:/scrtach/mvdemomaps/. Cache root directory will be set to  
default root directory.  
Aug 23, 2007 10:56:03 AM oracle.lbs.mapcache.config.CacheInstanceConfig loadCacheStorageDef  
WARNING: Invalid cache root directory:/scrtach/mvdemomaps/. Cache root directory will be set to  
default root directory.  
Aug 23, 2007 10:56:03 AM oracle.lbs.mapcache.MCSServlet\$ColdStart run  
**INFO: \*\*\* Oracle MapCacheServer started. \*\*\***

What you see in the above, is the initialization logs from MapViewer. It contains various informative logs such as where the config file is being read, what is the default screen resolution mapviewer will be using, Non-Spatial data providers being registered, as well as information about all the permanent data sources that you defined in the config file. You will also see the 3 bolded lines, which essentially means all the components of the MapViewer server has been properly initialized and ready for work. You may see some warnings about "Invalid cache root directory". This basically means that MapViewer cannot find the directory specified in a Oracle Maps map tile cache instance's definition (which is stored in the database view user\_sdo\_cached\_maps) on the box where MapViewer is running. You can ignore this warning as MapViewer will simply pick a default directory to store all the map tile files when they are first requested.

A typical MapViewer log record contains two lines. Such as the following:

```
Aug 23, 2007 10:56:03 AM oracle.lbs.mapcache.MCSServlet$ColdStart run
```

```
INFO: ** Oracle MapCacheServer started. **
```

The first line contains the timestamp and the routine that originates the log message, in this case it's the method run() of the class oracle.lbs.mapcache.MCSServlet\$ColdStart. The second line contains the actual log message itself, prefixed with its logging level. In this case the log message is simply "**\*\* Oracle MapCacheServer started.\*\***", with INFO being the logging level associated with this log record. If you set the MapViewer logging level to anything higher, such as WARN or ERROR (as would be the case for a production deployment), then you will not see this log record, because the level INFO is lower than the level WARN or ERROR.

## Logs generated during map rendering

The most useful log records are those generated when a map request is received by MapViewer and each theme in the request is being prepared and rendered. It is in these log records that you can spot potential problems, performance bottlenecks, as well as any mis-configured or invalid spatial data as well as mapping metadata such as styles/themes/basemaps. Lets now look at the logs generated for a typical (XML) map request. Again, in order to get the complete view of what's going on when MapViewer renders a map, it is crucial that you have set the logging level to **FINEST** in the mapViewConfig.xml file.

The map rendering related logs can be further divided into two categories: those that are produced for each individual theme, and the ones that related to the overall process or status of a map request. Because of the parallel nature, you often see the logs for the themes coming from a single map request get inter-mixed with each other. And you may also see logs from different map requests (in the case these map requests were received concurrently) intermixed.

### Theme logs

The following is a typical log record when MapViewer processes a pre-defined theme:

```
Sep 12, 2007 8:21:57 AM oracle.sdovis.theme.PredGeomThemeProducer loadFeaturesInWindow
FINEST: Theme query [ THEME_DEMO_HIGHWAYS ]: SELECT ROWID, GEOM, 'L.PH',
ROUTEN, 'M.SHIELD1', 1, 'rule#0' FROM INTERSTATES WHERE MDSYS.SDO_FILTER(GEOM,
MDSYS.SDO_GEOMETRY(2003, 8307, NULL, MDSYS.SDO_ELEM_INFO_ARRAY(1, 1003, 3),
MDSYS.SDO_ORDINATE_ARRAY(:mvqboxx1, :mvqboxy1, :mvqboxxh, :mvqboxyh)),
'querytype=WINDOW') = 'TRUE'
```

The first line here indicates which method produces this log record, in this case it's the loadFeaturesInWindow() method of the class PredGeomThemeProducer. This is just for your information. The second line (broken into multiple lines for better readability) is far more useful, as it not only lists the name of the theme being processed (THEME\_DEMO\_HIGHWAYS), but also the exact SQL query generated by MapViewer and sent to the database to fetch the data for this theme. You will see the columns being selected, and also the SDO\_FILTER predicate added by MapViewer to get only those features that are within the current map window. Note that the actual map window coordinates (minx,miny,maxx,mxy) are not displayed in this log record. These coordinates are found in a log record like the following (preceding the above record in the log file):

```
Sep 12, 2007 8:21:57 AM oracle.sdovis.theme.PredGeomThemeProducer loadFeaturesInWindow
FINEST: [ THEME_DEMO_HIGHWAYS ]: -122.7615,37.1516,-121.7615,37.9016
```

MapViewer also logs how long it takes for the databases to execute the query, and (more importantly) how long it takes to actually bring the result data from database to the middle tier where MapViewer is running. Such information can be found in log records like these:

```
FINER: [ THEME_DEMO_HIGHWAYS ] sql exec time: 585ms, total time loading 8 features: 2679ms.
```

As you can see, the above query took 585 milli-seconds (about half a second) to execute, but took 2679ms (almost 3 seconds) to actually fetch (load) the data produced by the query! This is actually not surprising considering spatial data are typically large volume so it takes longer to transmit them over a network than your average OLTP query results.

When you submit multiple themes in a map request, or if the map request contains a base map that in turn includes multiple themes, you will often see the above log records intermixed for different themes. This is resulted from the multi-threaded nature of how MapViewer processes themes. As the themes are processed in parallel of each other, the log records will be written to the file in no particular order among the themes.

MapViewer also tells you how long it took to render the features of a theme, such as:

```
FINER: time to render theme THEME_DEMO_HIGHWAYS with 8 styled features: 72ms
```

If the theme has labeling turned on, you will also see how long it took to label its features:

```
FINER: time to label theme THEME_DEMO_HIGHWAYS with 8 styled features: 112ms
```

Typically when you see something wrong in your application, it is one of the themes that are not defined or configured correctly. And you should look for an error or warning log for that theme in the log file. For instance, the following log records indicate a problem with a dynamically defined theme's query (note the error ORA-00904 which originates from the database server):

```
Sep 7, 2007 9:54:07 AM oracle.sdovis.theme.DynGeomThemeProducer prepareData
SEVERE: Error fetching data.
java.sql.SQLException: ORA-00904: "GEOMETRY": invalid identifier

    at oracle.jdbc.driver.SQLStateMapping.newSQLException(SQLStateMapping.java:76)
    at oracle.jdbc.driver.DatabaseError.newSQLException(DatabaseError.java:105)
    at oracle.jdbc.driver.DatabaseError.throwSQLException(DatabaseError.java:168)
    at oracle.jdbc.driver.T4CTTioer.processError(T4CTTioer.java:472)
    at oracle.jdbc.driver.T4CTTioer.processError(T4CTTioer.java:422)
    at oracle.jdbc.driver.T4C8Oall.receive(T4C8Oall.java:1015)
    at oracle.jdbc.driver.T4CPreparedStatement.doOall8(T4CPreparedStatement.java:204)
    at
oracle.jdbc.driver.T4CPreparedStatement.executeForDescribe(T4CPreparedStatement.java:857)
    at
oracle.jdbc.driver.T4CPreparedStatement.executeMaybeDescribe(T4CPreparedStatement.java:941)
    at oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout(OracleStatement.java:1280)
    at
oracle.jdbc.driver.OraclePreparedStatement.executeInternal(OraclePreparedStatement.java:3527)
    at
oracle.jdbc.driver.OraclePreparedStatement.executeQuery(OraclePreparedStatement.java:3579)
    at
oracle.jdbc.driver.OraclePreparedStatementWrapper.executeQuery(OraclePreparedStatementWrapper.java:267)
    at oracle.sdovis.theme.DynGeomThemeProducer.prepareData(DynGeomThemeProducer.java:332)
    at oracle.sdovis.Theme.prepareData(Theme.java:182)
    at oracle.sdovis.LoadThemeData.run(LoadThemeData.java:74)
Sep 7, 2007 9:54:07 AM oracle.sdovis.LoadThemeData run
SEVERE: Exception fetching data for theme JDBC_QUERY1.
```

## Map request logs

In addition to the log records produced for each individual theme, MapViewer also summarizes the total (wall-clock) time it took to load all the data from all the themes:

```
Sep 12, 2007 8:22:04 AM oracle.sdovis.DBMapMaker renderEm
```

INFO: \*\*\*\* time spent on loading features: 6685ms.

and the total rendering time (again, wall clock based):

INFO: \*\*\*\* time spent on rendering: 747ms

Note that above actually includes both the rendering and labeling time of all the themes.

Finally, depending on the output format requested, MapViewer also logs the "packing" time which is the time spent to convert an in-memory, temporary map image into final format (PNG, JPEG et al):

FINER: [RealWorker] packing time: 161ms

as well as the overall time spent on processing the map request itself (from parsing its XML document, to processing and rendering each theme, to generating the final output):

Sep 12, 2007 8:22:05 AM oracle.lbs.mapserver.core.RealWorker generateMapImage

FINER: [RealWorker] ----- total time: 10299ms